ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA INFORMÁTICA

GRADO EN INGENIERÍA DE COMPUTADORES

DESARROLLO DE APLICACIÓN MÓVIL PARA LA GESTIÓN DE EVENTOS DEPORTIVOS

Mobile Application Development for Managing Sports Events

Realizado por
JORGE ANTONIO BAUTISTA BENÍTEZ
Tutorizado por
GABRIEL JESÚS LUQUE POLO
JAMAL TOUTOUH EL ALAMIN
Departamento
LENGUAJES Y CIENCIAS DE LA COMPUTACIÓN

UNIVERSIDAD DE MÁLAGA
MÁLAGA, DICIEMBRE 2016

Fecha defensa:
El Secretario del Tribunal

Resumen: La idea de este proyecto es la creación de una aplicación móvil que permita a los usuarios la gestión de diferentes eventos deportivos. Para ello, los usuarios deben de darse de alta en la plataforma. Una vez se han dado de alta en la aplicación, los usuarios podrán realizar la gestión de los eventos deportivos, es decir, tendrán los permisos necesarios para poder crear un evento, modificarlo o cancelarlo siempre que lo deseen. Cuando el usuario crea un evento, debe indicar los participantes que van a asistir a dicho evento para que la aplicación avise a los mismos. La invitación a participar al evento se realizará mediante un correo electrónico para que los usuarios participantes indiquen si van a asistir al evento. Los usuarios participantes no necesitan estar registrados en la aplicación para poder confirmar su asistencia, pero únicamente podrán realizar esa acción, no dispondrán de los permisos necesarios para realizar la gestión de eventos.

Palabras clave: aplicación móvil, creación y gestión de eventos, AngularJS, Spring, Hibernate.

Abstract: The idea of this project is the creation of a mobile application that allows users to manage different sport events. For this, the users must be registered in the platform. Once they have registered in the application, users will be able to manage the sport events, they will have the necessary permissions to be able to create an event, modify it or cancel it whenever they wish. When the user creates an event, they must indicate the participants who will attend the event so that the application notifies them. The invitation to participate to the event will be made by means of an email so that the participating users indicate if they are going to attend the event. Participant users do not need to be registered in the application in order to confirm their attendance, but they will only be able to do that, they will not have the necessary permissions to carry out the event management.

Keywords: mobile application, create and event management, AngularJS, Spring, Hibernate.

Índice

Capítulo 1 Introducción

En la actualidad, una gran parte de la población dedica parte de su tiempo de ocio a la práctica de deporte. Esto se debe principalmente a que desde las instituciones sanitarias se insiste en los beneficios del mismo. Así, han proliferado multitud de establecimientos que se encargan de que las personas realicen ejercicio físico en ellos, ya sean gimnasios o recintos deportivos. Aunque estos recintos ofrecen servicios de actividades programadas, no es raro que los propios usuarios utilicen dichas infraestructuras para organizar distintos eventos deportivos de equipo que ellos gestionan, como pueden ser partidos de fútbol, de baloncesto o voleibol. Sin embargo, durante la organización de dichos eventos pueden surgir dificultades tales como: encontrar el número suficiente de usuarios para completar los equipos, localizar el lugar en el que se pueda organizar el evento, coordinarse a la hora de proporcionar el material necesario para el partido, etc. En este Trabajo Final de Grado (TFG) se quiere desarrollar una aplicación móvil que asista a los usuarios que quieren organizar o participar en un evento deportivo de estas características.

A lo largo de los últimos años, las tecnologías de la información y la comunicación (TIC) han experimentado una gran evolución. Un ejemplo claro de ello es la evolución de la tecnología móvil, actualmente empleando dispositivos móviles celulares se pueden realizar una gran cantidad de tareas, antes impensables. Hoy en día puedes establecer vínculos afectivos con personas que están a kilómetros de distancia mediante las redes sociales como Facebook o Twitter; se puede realizar la compra utilizando aplicaciones como las ofrecidas por Carrefour, El Corte Inglés o DIA; e incluso se pueden realizar pagos en locales comerciales usando el móvil alguna de las plataformas como BBVA Wallet, Caixabank Pay y Bankinter Pagos TVM. Esto es solo una representación de la multitud de tareas que se pueden realizar usando un dispositivo móvil o tableta.

Así pues, también existen aplicaciones móviles con las que los usuarios pueden realizar la gestión de diferentes eventos deportivos, una de las más conocidas es RosterBot [1]. Esta plataforma ofrece la oportunidad de que los usuarios puedan organizar equipos en diferentes deportes. De esta manera, el sistema invita a los participantes para que estos decidan si acuden al evento y ofrece datos como fechas, perfiles de jugadores y ubicaciones. Es una aplicación desarrollada principalmente para participar en eventos con participantes que no se encuentran dentro del entorno del usuario.

Es una aplicación muy completa, pero ofrece algunas carencias. La interfaz de esta plataforma no es muy buena, no es intuitiva ni predecible y de esta manera no puede haber una correcta interacción con el usuario. Es cierto que no existe una interfaz válida para todos los usuarios, pero también hay que tener en cuenta que la interfaz debería ser lo más accesible e intuitiva para que cualquier usuario pueda interactuar de la manera más sencilla posible. Otro problema que tiene la aplicación Rosterbot es que únicamente está diseñada como plataforma web, lo que restará accesibilidad y usabilidad a aquellos usuarios que deseen acceder mediante el teléfono móvil.

Otra aplicación muy conocida en nuestro país es Padelon [2]. Esta aplicación es usada principalmente entre aquellos usuarios que practican el deporte del Pádel, funciona como una red social de jugadores de pádel que permite a los usuarios: crear partidos, participar en diferentes partidos y además puede restringir el acceso a los partidos por el nivel de sus jugadores. El problema de este tipo de aplicación es que únicamente se centra en un deporte, en este caso el pádel, y si el usuario desea practicar cualquier otro deporte tendrá que volver a descargar otra aplicación que le permita gestionar otro evento con la actividad deportiva deseada.

En este documento se va proponer una manera de gestionar eventos deportivos, similar a la aplicación expuesta anteriormente, mediante una aplicación móvil para poder facilitar a los diferentes usuarios la práctica de diferentes deportes con otros participantes.

En este TFG proponemos el desarrollo de Esportvent. Esta plataforma va a ofrecer a los usuarios la capacidad de organizar eventos deportivos, pudiendo gestionar multitud de opciones (algunas no contempladas por RosterBot y Padelon): la ubicación de donde se va a desarrollar, el coste de la actividad, los participantes al evento con sus posiciones definidas o no, material necesario y quien lo va a proporcionar, etc. Una de las principales ventajas de esta aplicación es que va a permitir enviar correos electrónicos a todos los usuarios participantes, estén registrados o no, para que confirmen la asistencia al evento en la plataforma, esto va a permitir que la aplicación se difunda también a los usuarios que no la conozcan, facilitando su penetración en el mercado.

1.1 Objetivos

El objetivo principal de este TFG es el desarrollo de Esportvent, una aplicación móvil con la que los usuarios puedan realizar la gestión de eventos deportivos.

La aplicación tiene dos grandes grupos de funcionalidades, la administración de usuarios (alta, control de acceso o actualización de sus datos) y la gestión de eventos (creación, edición y eliminación de eventos).

1.2 Estructura de la memoria

La memoria del presente TFG se divide en los capítulos que se van a describir a continuación.

- El segundo capítulo profundiza en las principales herramientas y tecnologías utilizadas para el desarrollo del trabajo.
- El tercer capítulo realiza un pequeño resumen de este TFG, se realiza una descripción del funcionamiento de la aplicación junto a la metodología usada en el proyecto. Además se especifica la planificación desarrollada en el mismo.
- El cuarto capítulo se especifica el diseño de la aplicación especificando los posibles casos de uso que se pueden producir en la plataforma y los diferentes requerimientos funcionales y no funcionales.
- El quinto capítulo profundiza en la implementación de la aplicación junto con los diagramas de base de datos y UML de las clases de dicha aplicación.
- El sexto capítulo especifica las conclusiones que se han obtenido mientras se ha desarrollado la aplicación

Capítulo 2. Tecnologías y herramientas de desarrollo

Este capítulo describe las tecnologías y herramientas usadas para la realización del proyecto. Se eligieron estas herramientas debido a su flexibilidad a la hora de desarrollar gran variedad de aplicaciones y también quería aprender a desarrollar con este tipo de tecnologías ya que son muy utilizadas en la actualidad y con muchas de ellas no había tenido contacto antes del proyecto.

Las tecnologías usadas en la aplicación se pueden subdividir en cuatro grupos. Las utilizadas para el desarrollo de la aplicación servidor, las utilizadas para el desarrollo de la aplicación cliente, tecnologías relacionadas con el manejo de datos y herramientas usadas para el desarrollo.

- **Tecnologías de la aplicación servidor:**

En su conjunto la aplicación servidora se basa en una implementación Java, usando diferentes *frameworks* que nos definirán la arquitectura y que nos facilitarán un resultado muy estructurado. Las tecnologías usadas en el desarrollo de la parte servidora son las siguientes:

- Lenguaje de programación Java J2EE.
- Gestor de proyectos Maven [3]
- Spring Framework [4] para la inyección de dependencias entre capas.
- Spring Security Framework [5] para la gestión de autentificación.
- Web Services Rest para proveer de servicios a la aplicación cliente.
- Servidor de aplicaciones Apache Tomcat.

- **Tecnologías de la aplicación cliente:**

En su conjunto la aplicación cliente se basa en la implementación de un cliente ligero en JavaScript [6] para que pueda ser ejecutado mediante el motor de navegación de cualquier Smartphone. Las tecnologías usadas en el desarrollo de la parte cliente son las siguientes:

- HTML 5 [7]
- JavaScript y JQuery.
- CSS y Bootstrap para la maquetación *responsive* de la aplicación.
- AngularJS [8] Framework para la implantación de la arquitectura de la aplicación cliente y la gestión de llamadas a Web Services REST [9] con JSON.

- **Tecnologías relacionadas con el manejo de datos**

Las diversas tecnologías que se han usado que están relacionadas con el manejo de datos son las siguientes:

- Base de datos MySQL.
- Hibernate Framework para el desarrollo de la capa de persistencia (DAO).
- JSON como formato de comunicación con la aplicación cliente.

- **Herramientas usadas para el desarrollo**

Las herramientas usadas para la realización de este proyecto son:

- Herramienta de desarrollo Eclipse.
- Herramienta de desarrollo de arquitectura de base de datos MySQL Workbench.
- Herramienta de control de versiones Git.
- ObjectAid para la realización de diagramas UML

En las siguientes subsecciones se introducen las tecnologías más importantes de las expuestas anteriormente.

2.1 MySQL

MySQL es un popular sistema de gestión de bases de datos relacional, multihilo y multiusuario con más de seis millones de instalaciones. Es de código abierto e implementa un amplio conjunto de funciones del lenguaje SQL.

2.2 Hibernate

Hibernate es una herramienta de mapeo objeto-relacional (ORM) para Java que facilita el mapeo de atributos entre una base de datos relacional tradicional y el modelo de objetos de una aplicación, mediante archivos declarativos (XML) o anotaciones en los *beans* de las entidades que permiten establecer estas relaciones. Hibernate es *software* libre, distribuido bajo los términos de la licencia GNU LGPL. Convertirá los datos entre los tipos utilizados por Java y los definidos por SQL. Está diseñado para ser flexible en cuanto al esquema de tablas utilizado, para poder adaptarse a su uso sobre una base de datos ya existente. También tiene la funcionalidad de crear la base de datos a partir de la información disponible.

2.3 Spring Framework

Spring *framework* se compone de varios módulos, todos ellos giran en torno a *Spring Core* el cual hace uso intensivo del patrón de inyección de dependencias. El contenedor de Spring es uno de los puntos centrales de Spring, se encarga de crear los objetos, conectarlos entre sí, configurarlos y además controla los ciclos de vida de cada objeto

mediante el patrón de inyección de dependencias. Se puede personalizar el contenedor de Spring mediante configuración XML o programáticamente mediante anotaciones.

Algunas de las cosas que tendremos que configurar a la hora de crear un Contexto de aplicación de Spring son:

- Crear *beans* individualmente.
- Configurar los servicios que usará.

Los *beans* son clases que cumplen ciertas propiedades como tener constructor por defecto, acceso a los atributos, etc. Además se pueden declarar mediante anotaciones en POJO (*Plain Old Java Object*, objetos de Java) o mediante XML.

2.4 Spring Security

Spring Security proporciona servicios de seguridad para aplicaciones de *software* empresariales basados en J2EE, enfocado particularmente sobre proyectos construidos usando Spring Framework.

2.5 JSON

JSON, acrónimo de JavaScript *Object Notation*, es un formato ligero para el intercambio de datos. JSON es un subconjunto de la notación literal de objetos de JavaScript que no requiere el uso de XML.
JSON se emplea habitualmente en entornos donde el tamaño del flujo de datos entre cliente y servidor es de vital importancia cuando la fuente de datos es explícitamente confiable.

2.6 Web Services REST

REST es una familia de arquitecturas. Cualquier arquitectura de servicios distribuidos que cumpla con una serie de requisitos se puede considerar como una arquitectura REST. Este servicio se basa en el principio de CRUD (*Create, Read, Update, Delete*) para realizar sus operaciones.

2.7 JQuery

JQuery es una biblioteca de JavaScript, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. JQuery es la biblioteca de JavaScript más utilizada.

2.8 AngularJS

AngularJS [10] es un *framework* MVC (Modelo Vista Controlador) de JavaScript para el desarrollo *Front End* de aplicaciones web del tipo SPA (*Single-Page Applications*). Los componentes que componen angular son:

- **Views:** se realizan mediante HTML. Cada control tiene uno o más *bindings* contra el *controller* que define un alcance o *$scope* con un juego de variables que hace de modelo
- **Controller:** adapta la vista con el modelo final, habla con los *services/factories* para manejar la comunicación con los orígenes de datos
- **Services:** reutilizan funcionalidades comunes entre *controllers*, tales como obtener o actualizar información desde los orígenes de datos locales o remotos, pueden manejar niveles de cache, etc.
- **Module:** funciona como contenedor de elementos visuales, lógica de la vista y manejo del comportamiento del lado del cliente.
- **Route:** relaciona un servicio REST con una funcionalidad en el cliente.

2.9 Bootstrap

Bootstrap [11] es un *framework* originalmente creado por Twitter, que permite crear interfaces web con CSS [12] y JavaScript, cuya particularidad es la de adaptar la interfaz del sitio web al tamaño del dispositivo en que se visualice. Es decir, el sitio web se adapta automáticamente al tamaño de una PC, una Tablet u otro dispositivo. Esta técnica de diseño y desarrollo se conoce como "*responsive design*" o diseño adaptativo.

2.10 Maven

Maven es una herramienta de *software* para la gestión y construcción de proyectos Java. Tiene un modelo de configuración de construcción basado en un formato XML. Maven utiliza un *Project Object Model* (POM) para describir el proyecto de *software* a construir, sus dependencias de otros módulos y componentes externos, y el orden de construcción de los elementos. Viene con objetivos predefinidos para realizar ciertas tareas claramente definidas, como la compilación del código y su empaquetado.

.

Capítulo 3. Visión general del proyecto y metodología

En este capítulo se va a presentar una visión general del TFG teniendo en cuenta los objetivos que se persiguen en el proyecto, el funcionamiento de la aplicación, la metodología que se ha llevado a cabo y la planificación que se ha seguido para el desarrollo del mismo.

3.1 Funcionamiento de Esportvent

Este TFG trata el desarrollo de una aplicación para la organización de eventos deportivos. La aplicación tendrá dos grandes apartados: i) uno para la creación y gestión de un evento y ii) la inclusión de participantes al evento. A esto lo acompaña la gestión de usuarios.

Para poder realizar estas funciones los usuarios interaccionarán con la plataforma que implementa Esportvent. Inicialmente el usuario que va a crear o gestionar un evento deberá registrarse empleando la interfaz de la aplicación cliente. Así se recopilará la información y podrá ser enviada a nuestra aplicación servidora para su posterior tratamiento y análisis (ver Figura 1). El usuario que ya se ha registrado podrá realizar la creación de eventos en diferentes actividades deportivas, además de añadir a los participantes que deseen participar en el evento

Cuando el usuario añade en el evento a los usuarios participantes, la aplicación les enviará un correo electrónico para que cada uno de los participantes acepte o decline la invitación al evento. Estos usuarios no tienen que estar registrados en la aplicación, aunque en el correo se les invitará a que puedan hacerlo mediante un enlace de acceso a la plataforma. Si algún participante rechaza la asistencia al evento, la aplicación informará al usuario creador del evento por correo electrónico para que pueda invitar a otro asistente a la actividad (ver Figura 1).

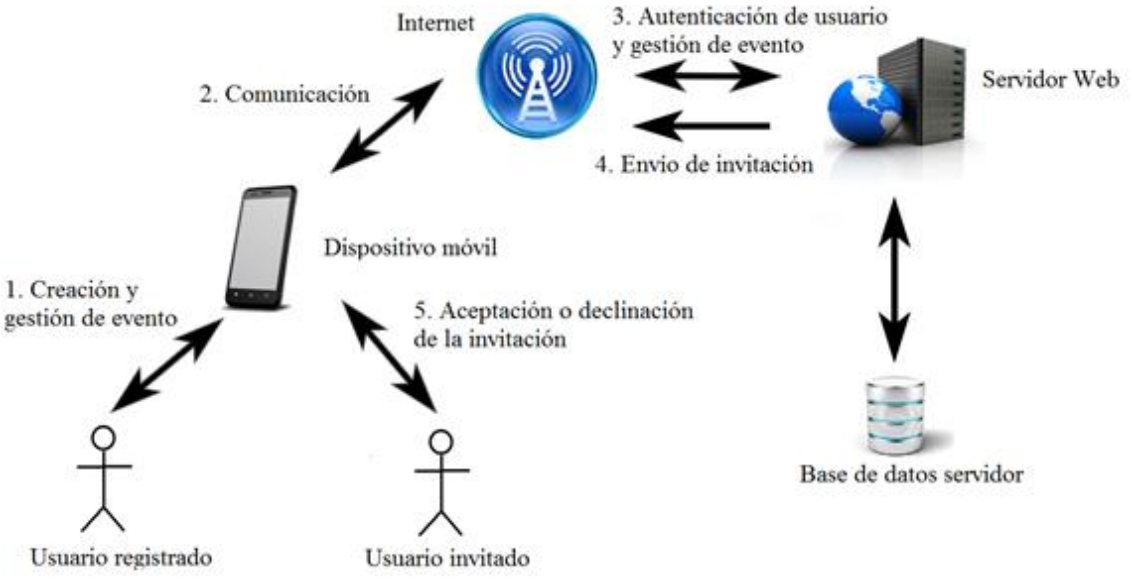


Figura 1. *Diagrama de funcionamiento.*

3.2 Metodología

En esta sección se resume la metodología empleada en el desarrollo del TFG. Para la realización del sistema se ha dividido el mismo en dos: i) una aplicación cliente y ii) una aplicación servidora. La metodología de ingeniería del *software* que se ha seguido en la aplicación va a ser "en cascada".

La aplicación cliente basa en el patrón de diseño MVC donde el lenguaje de programación principal será JavaScript empleando AngularJS [13]. Las vistas se han implementado usando HTML y CSS. La aplicación servidora se ha desarrollado principalmente en Java junto con los *frameworks* Spring e Hibernate. Finalmente, para poder comunicar la aplicación cliente con la aplicación servidora de manera eficiente, se han usado servicios REST que transmiten la información empleando JSON.

3.3 Planificación

El proyecto está dividido en cuatro fases principales y comprende las 296 horas exigidas para su realización. Las cuatro fases en las que se ha dividido el TFG han sido las siguientes: "Investigación", "Análisis de requisitos y diseño", "Implementación" y "Pruebas y documentación" del trabajo fin de grado.

- Investigación (54 horas)
    - Análisis de bibliotecas de terceros para la comunicación cliente-servidor mediante tecnologías REST.
    - Estudio de herramientas de tratamiento de datos.
    - Estudio de *frameworks* para desarrollo multiplataforma de aplicaciones móviles.
- Análisis de requisitos y diseño (73 horas)
    - Análisis de requisitos.
    - Especificación de casos de uso.
    - Modelado de la aplicación móvil.
    - Diseño de la base de datos (Esquema DDL).
    - Diseño de la API REST del servidor.
    - Diseño de la interfaz de usuario de la aplicación móvil.
- Implementación (94 horas)
    - Desarrollo de la aplicación móvil.
    - Desarrollo y despliegue del servidor de aplicaciones móviles.
    - Despliegue de la base de datos.
- Pruebas y documentación (75horas)
    - *Testing* de la aplicación móvil (pruebas unitarias).
    - Revisión del código a nivel de documentación.
    - Redacción de la memoria del TFG.

Capítulo 4. Análisis y modelado de requisitos

En este capítulo se va presentar la metodología empleada para el análisis de los requisitos y el modelado de los casos de uso.

4.1 Metodología de desarrollo

La formalización del proceso de desarrollo se define como un marco de referencia denominado ciclo de desarrollo del *software* o ciclo de vida del desarrollo del *software*. Se puede describir como, "el período de tiempo que comienza con la decisión de desarrollar un producto *software* y finaliza cuando se ha entregado éste".

El ciclo de desarrollo *software* se utiliza para estructurar las actividades que se llevan a cabo en el desarrollo de un producto *software*. Siendo útil para la comprensión y el control del proceso.

Seguiré el modelo clásico, técnica rígida para mejorar la calidad y reducir los costos del desarrollo de *software*. Tradicionalmente es conocido como "modelo en cascada", porque su filosofía es completar un paso con un alto grado de exactitud, antes de empezar el siguiente. Donde destacan diversas etapas:

- Análisis de Requisitos: Elaborar una especificación completa y validada de las funciones requeridas del producto de *software*.

- Modelado: se formaliza en modelos que describen completamente el sistema la especificación de requisitos obtenida en el punto anterior.

- Diseño: Elaborar una especificación completa y validada de la arquitectura global *hardware-software*, de la estructura de control y de la estructura de datos del producto, así como un esquema de los manuales de usuarios.

- Implementación: Donde transformaremos el modelo de diseño en código fuente, es decir, haremos funcionar el sistema global *hardware-software* incluyendo conversión de programas y datos, instalación y capacitación.

- Prueba: encontraremos diferencias entre el comportamiento esperado del sistema y el comportamiento observado en la ejecución del sistema implementado.

La elección del modelo de desarrollo clásico frente al modelo iterativo se ha llevado a cabo debido a que el proyecto es desarrollado por una sola persona, dando un mayor control y conocimiento sobre lo que se realiza en cada etapa, sin tener mucho sentido una metodología de iteraciones, aunque debe añadirse que en la actualidad el modelo de iteraciones es el más utilizado

Obviamente antes de todas esas fases, viene una etapa de aprendizaje y familiarización de las tecnologías que desconocía.

4.2 Requerimientos funcionales

RF.1. El sistema debe permitir la existencia del tipo de usuario registrado.

1.1. Los usuarios registrados podrán acceder a todos los componentes.

1.2. Un usuario registrado cuya cuenta haya sido desactivada podrá volver a reactivar la cuenta volviendo a acceder a la aplicación desde la pantalla de acceso.

RF.2. Deberá existir un sistema de autentificación de usuarios registrados que les dé acceso a los componentes de la aplicación.

RF.3. El sistema deberá permitir la posibilidad de desactivación de cuentas.

3.1. Un usuario registrado podrá desactivar su cuenta cuando desee. Aunque un usuario desactive su cuenta sus eventos permanecerán en su cuenta inactiva hasta que el usuario desee reactivarla. La cuenta podrá ser reactivada cuando el usuario desee.

RF.4. El sistema deberá permitir la solicitud de un recordatorio de contraseña si ésta ha sido olvidada.

RF.5. El sistema deberá mostrar una página donde se explique su funcionamiento completo. Dicha página debe ser visible para cualquier tipo de usuario.

RF.6. Durante el alta de un evento, deberá indicarse el periodo de fechas y la franja horaria diaria en la que se desea realizar la actividad.

RF.7. Deben poder actualizarse los datos de la cuenta y los datos del evento.

RF.8. Debe existir un método para poder enviar incidencias al administrador del sistema.

RF.9. La dirección donde se realiza el evento podrá ser geo –posicionada en un mapa de manera visual. Conjunta a esta la funcionalidad, deberá existir una función “Ir ahora” que permita con una sola pulsación guiar al usuario hasta el evento en el que va a participar, utilizando para ello el GPS del terminal y cualquier aplicación de navegación GPS que el terminal tenga instalada. Esta funcionalidad deberá estar disponible desde el listado de la búsqueda de eventos participantes.

RF.10. Gestión de evento: un usuario podrá crear, modificar o cancelar los eventos deportivos que desee. Cada vez que realice cualquier tipo de gestión se notificará a los participantes mediante correo electrónico.

4.3 Requerimientos no funcionales

RNF.1. No se permitirá acceso a ningún recurso, excepto a la pantalla de información, sin autenticación previa

RNF.2. No se permitirá que usuarios ajenos al sistema tengan acceso

RNF.3. El sistema informará del resultado de realizar cualquier acción a través de mensajes por pantalla o mediante correo electrónico.

RNF.4. El sistema debe ser fácilmente extensible. La aplicación debería de poder añadir nuevos módulos de funcionalidad

RNF.5. Se desarrollará un manual de instalación y uso dirigido a todo tipo de usuarios, ya sean usuarios más avanzados o de conocimientos básico de informática.

RNF.6. Esportvent no se hará responsable de posibles datos incorrectos introducidos en el sistema, ni de los perjuicios que de ellos puedan derivar. Solo los usuarios son responsables de la veracidad de los datos que introducen en el alta.

RNF.7. Esportvent no se hará responsable de posibles malas prácticas por parte de los usuarios.

4.4 Casos de uso

En esta sección se va a proceder a definir los posibles actores que pueden interactuar en la aplicación y los posibles casos de uso.

4.4.1 Definición de actores

Los distintos roles que un usuario puede tener dentro del sistema son los siguientes:

- **Usuario invitado:** es aquel usuario que visita el sistema sin iniciar ninguna sesión en el mismo como usuario registrado. Este usuario solo puede registrarse, autenticarse y visitar la sección de “modo de funcionamiento”.

- **Usuario registrado:** es aquel usuario que se ha identificado en la aplicación utilizando una cuenta que previamente ha creado en ella. El usuario registrado tiene acceso a todos los componentes.

4.4.2 Identificación de los casos de uso

Los casos de uso que se distinguen en nuestro proyecto los agruparemos en dos grupos: "gestión de cuentas y eventos" y "ayuda". Son los siguientes:

1. Gestión de cuentas y eventos:

1.1. Control de acceso.
1.2. Crear cuenta.
1.3. Actualizar cuenta.
1.4. Desactivar cuenta.
1.5. Recuperar contraseña.
1.6. Crear evento.
1.7. Actualizar evento.
1.8. Cancelar evento.

2. Ayuda:

2.1. Modo de funcionamiento.
2.2. Crear incidencia.

4.4.3. Diagrama de casos de uso

La Figura 2 muestra cómo quedaría el diagrama con los posibles casos de uso.

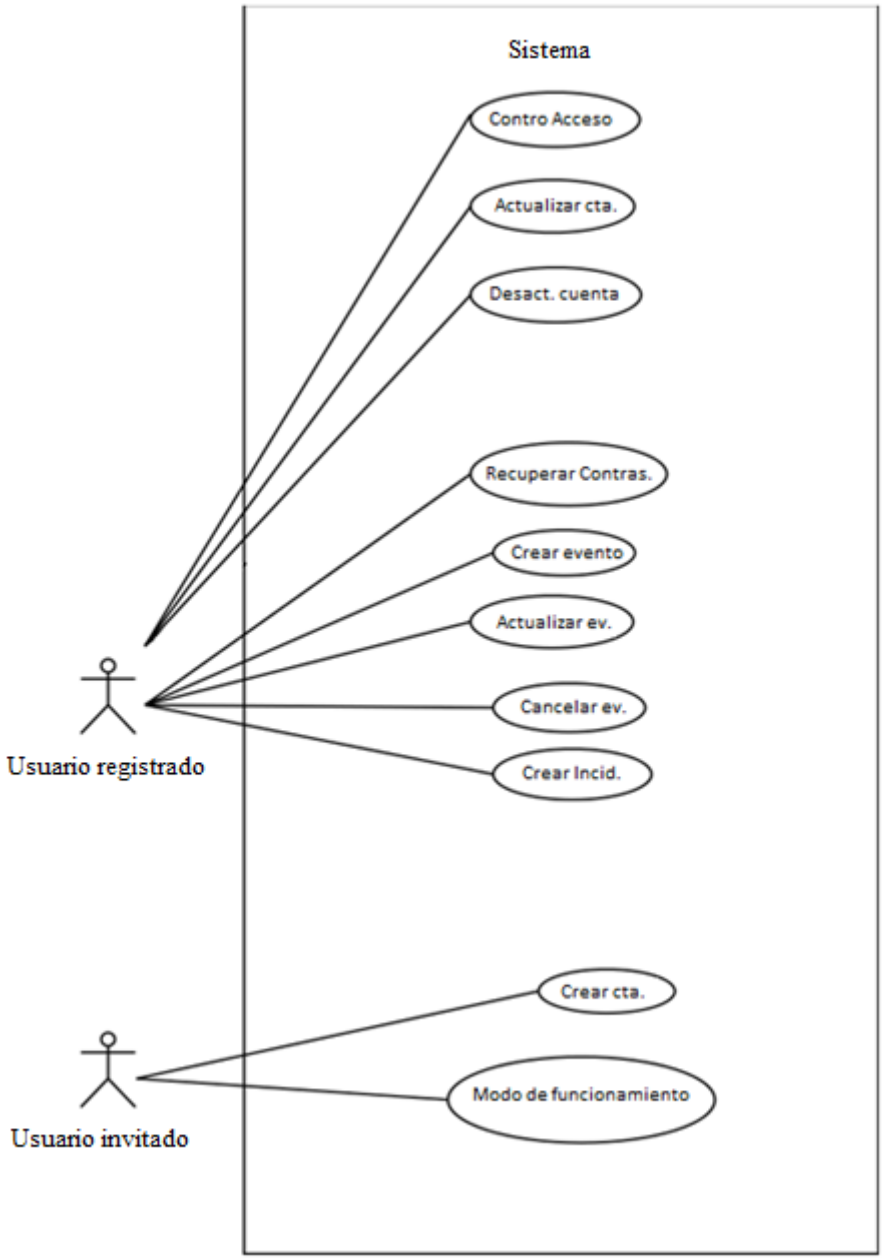


Figura 2. *Diagrama de casos de uso.*

4.4.4. Descripción de casos de uso

Los casos de uso que se han estudiado en este proyecto son los siguientes:

NOMBRE DEL CASO DE USO	Control de acceso	1.1
RESUMEN	Este caso de uso permite al usuario acceder a los componentes para usuarios registrados.	
DEPENDENCIAS		
ACTORES	Usuario registrado	
PRECONDICIONES	El usuario debe estar registrado en el sistema.	
POSTCONDICIONES		
CURSO NORMAL	1.- Este caso de uso comienza cuando el usuario accede a la aplicación. 2.- El usuario introduce su email y contraseña. 3.- El usuario pulsa "Acceder" 4.- Se accede a la zona restringida.	
CURSOS ALTERNATIVOS	4a.- El usuario no está registrado en el sistema, por lo que se le muestra un mensaje informándole de que no puede acceder.	
OBSERVACIONES		

NOMBRE DEL CASO DE USO	Crear cuenta	1.2
RESUMEN	Este caso de uso permite al usuario registrarse en el sistema.	
DEPENDENCIAS		
ACTORES	Usuario invitado	
PRECONDICIONES		
POSTCONDICIONES		
CURSO NORMAL	1.- Este caso de uso comienza cuando el usuario accede a la aplicación y pulsa "Registrarse" 2.- El usuario introduce todos los datos personales solicitados. 3.- El usuario pulsa "guardar". 4.- Se redirige a la pantalla de Acceso.	
CURSOS ALTERNATIVOS	4a.- El usuario no ha rellenado los datos correctamente, en tal caso, se marcan los campos como obligatorios o como mal validados.	
OBSERVACIONES		

NOMBRE DEL CASO DE USO	Actualizar cuenta	1.3
RESUMEN	Este caso de uso permite al usuario modificar los datos de su cuenta en el sistema.	
DEPENDENCIAS		
ACTORES	Usuario registrado	
PRECONDICIONES	El usuario debe haber iniciado la sesión en la aplicación.	
POSTCONDICIONES		
CURSO NORMAL	1.- Este caso de uso comienza cuando el usuario accede a la aplicación y pulsa "Actualizar sus datos" 2.- El usuario modifica los datos deseados. 3.- El usuario pulsa "guardar". 4.- Se redirige a la pantalla de "Selección de operaciones".	
CURSOS ALTERNATIVOS	4a.- El usuario no ha rellenado los datos correctamente, en tal caso, se marcan los campos como obligatorios o como mal validados.	
OBSERVACIONES		

NOMBRE DEL CASO DE USO	Desactivar cuenta	1.4
RESUMEN	Este caso de uso permite al usuario desactivar su cuenta en el sistema.	
DEPENDENCIAS		
ACTORES	Usuario registrado	
PRECONDICIONES	El usuario debe haberse *logeado* en la aplicación.	
POSTCONDICIONES	El usuario pasará a ser usuario no registrado. Los eventos asociados a la cuenta desactivada seguirán en la cuenta del usuario.	
CURSO NORMAL	1.- Este caso de uso comienza cuando el usuario accede a la aplicación y pulsa "Eliminar cuenta" 2.- La cuenta se desactiva. 3.- Se redirige a la pantalla de "*Login*".	
CURSOS ALTERNATIVOS		
OBSERVACIONES		

NOMBRE DEL CASO DE USO	Recuperar contraseña	1.5
RESUMEN	Este caso de uso permite al usuario recuperar su contraseña si la ha olvidado.	
DEPENDENCIAS		
ACTORES	Usuario registrado	
PRECONDICIONES	El usuario debe haber introducido durante su alta un email válido y real.	
POSTCONDICIONES	El usuario deberá revisar su correo electrónico, a la espera del mensaje recibido.	
CURSO NORMAL	1.- Este caso de uso comienza cuando el usuario accede a la aplicación y pulsa "Recuperar contraseña" 2.- El usuario introduce su email. 3.- El usuario pulsa "Recuperar". 4.- Un email le es enviado desde el sistema de forma automática con la información de su contraseña. 5.- Se redirige a la pantalla de "*Login*".	
CURSOS ALTERNATIVOS	4a.- Si el email no existe en el sistema, se informará con un mensaje de error.	
OBSERVACIONES		

NOMBRE DEL CASO DE USO	Crear evento	1.6
RESUMEN	Este caso de uso permite al usuario añadir un evento nuevo a su cuenta	
DEPENDENCIAS		
ACTORES	Usuario registrado	
PRECONDICIONES	El usuario debe estar registrado	
POSTCONDICIONES	Su evento podrá ser modificado o cancelado.	
CURSO NORMAL	1.- Este caso de uso comienza cuando el usuario accede a la aplicación y pulsa "Crear evento" 2.- El usuario introduce todos los datos requeridos del evento. 3.- Las coordenadas de evento son cargadas automáticamente. 4.- El usuario pulsa "guardar". 5.- El evento es almacenado. 6.- Se redirige a la pantalla de "Selección de operaciones".	
CURSOS ALTERNATIVOS	Si las coordenadas no pueden ser recuperadas (dirección), se mostrará un error y el evento no podrá ser guardado en el sistema.	

NOMBRE DEL CASO DE USO	Actualizar eventos	1.7
RESUMEN	Este caso de uso permite al usuario modificar los eventos asociados a su cuenta.	
DEPENDENCIAS		
ACTORES	Usuario registrado	
PRECONDICIONES	El usuario tiene que tener ya un evento asociado a su cuenta.	
POSTCONDICIONES	Su evento podrá ser modificado hasta 6h antes de la franja horaria indicada.	
CURSO NORMAL	1.- Este caso de uso comienza cuando el usuario accede a la aplicación y pulsa "Ver evento" 2.- El usuario modifica los datos deseados del evento 3.- El usuario pulsa "guardar evento". 4.- Se redirige a la pantalla de "Selección de operaciones".	
CURSOS ALTERNATIVOS		
OBSERVACIONES		

NOMBRE DEL CASO DE USO	Cancelar evento	1.8
RESUMEN	Este caso de uso permite al usuario cancelar el evento deseado asociado a su cuenta.	
DEPENDENCIAS		
ACTORES	Usuario registrado	
PRECONDICIONES	El usuario tiene que tener ya un evento asociado a su cuenta.	
POSTCONDICIONES	El evento permanecerá como "cancelado" en su cuenta	
CURSO NORMAL	1.- Este caso de uso comienza cuando el usuario accede a la aplicación y pulsa "Ver evento" 2.- El usuario selecciona el estado "Cancelado". 3.- El evento es cancelado. 4.- Se redirige a la pantalla de "Selección de operaciones".	
CURSOS ALTERNATIVOS		
OBSERVACIONES		

NOMBRE DEL CASO DE USO	Modo de funcionamiento	2.1
RESUMEN	Este caso de uso permite al usuario acceder a una página donde se explica el modo de uso de la aplicación.	
DEPENDENCIAS		
ACTORES	Usuario invitado	
PRECONDICIONES		
POSTCONDICIONES		
CURSO NORMAL	1.- Este caso de uso comienza cuando el usuario accede a la aplicación y pulsa “Modo de funcionamiento” 2.- Se mostrará un resumen del modo de funcionamiento de la aplicación.	
CURSOS ALTERNATIVOS		
OBSERVACIONES		

NOMBRE DEL CASO DE USO	Crear incidencia	2.2
RESUMEN	Este caso de uso permite al usuario enviar una incidencia al administrador del sistema.	
DEPENDENCIAS		
ACTORES	Usuario registrado	
PRECONDICIONES	El usuario debe estar registrado en el sistema.	
POSTCONDICIONES		
CURSO NORMAL	1.- Este caso de uso comienza cuando el usuario accede a la aplicación y pulsa “Crear incidencia” 2.- El usuario introduce un comentario con el contenido de la incidencia. Un número de incidencia es generado automáticamente. 3.- El usuario pulsa “enviar” 3.- La incidencia es enviada automáticamente al administrador del sistema.	
CURSOS ALTERNATIVOS		
OBSERVACIONES		

4.5 Operaciones del sistema

Las operaciones que se realizan a nivel de sistema son las que se van a representar en las figuras que van de la 3 a la 14:

1. Autenticarse.

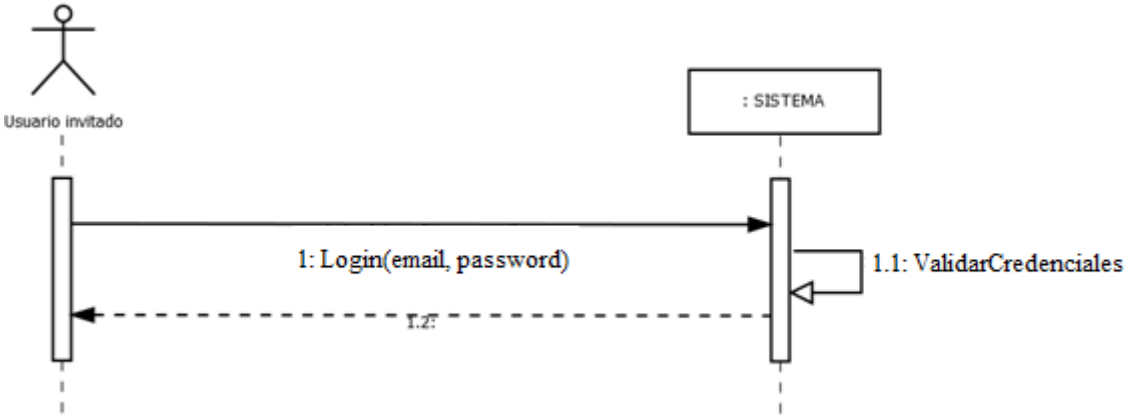


Figura 3. *Esquema autenticación.*

2. Modo funcionamiento.

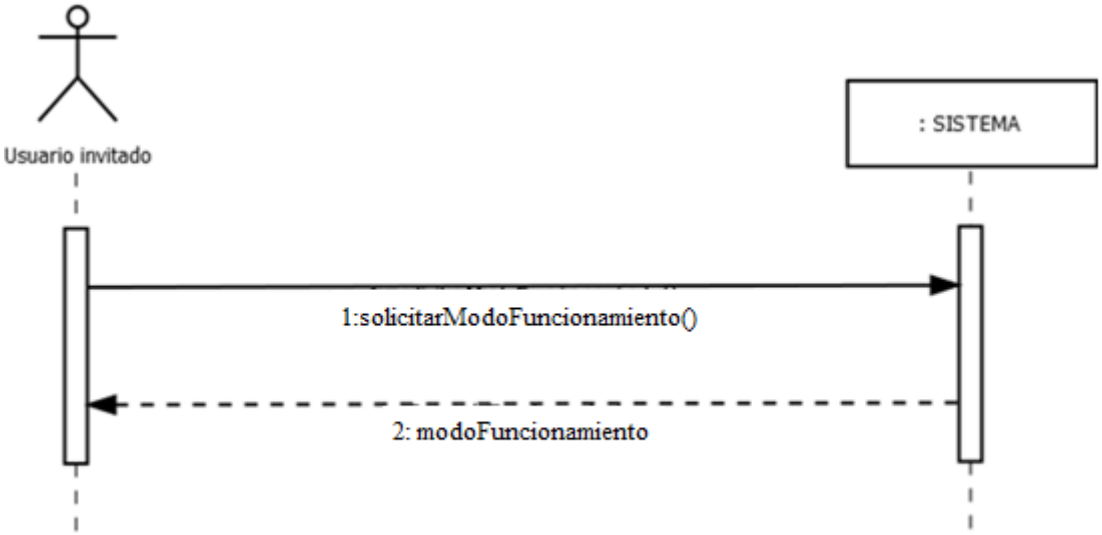


Figura 4. *Esquema del modo de funcionamiento.*

3. Creación cuenta.

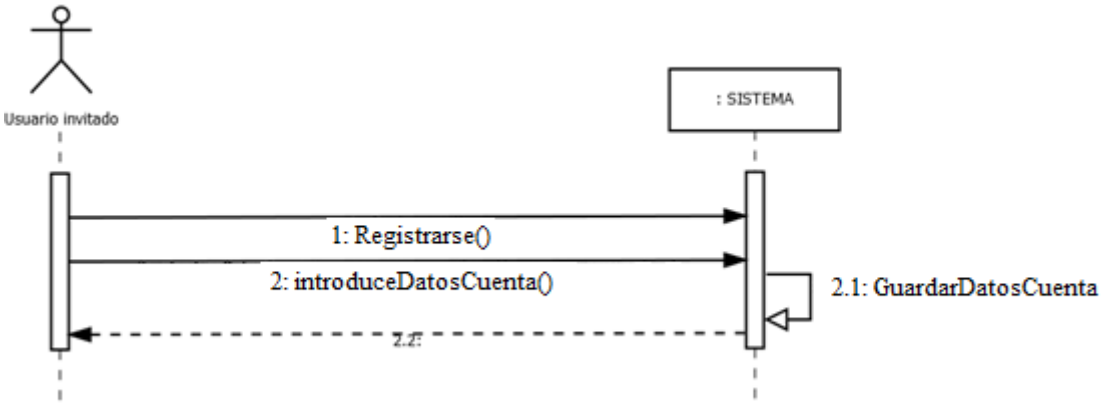


Figura 5. *Esquema de la creación de cuenta.*

4. Ver datos cuenta

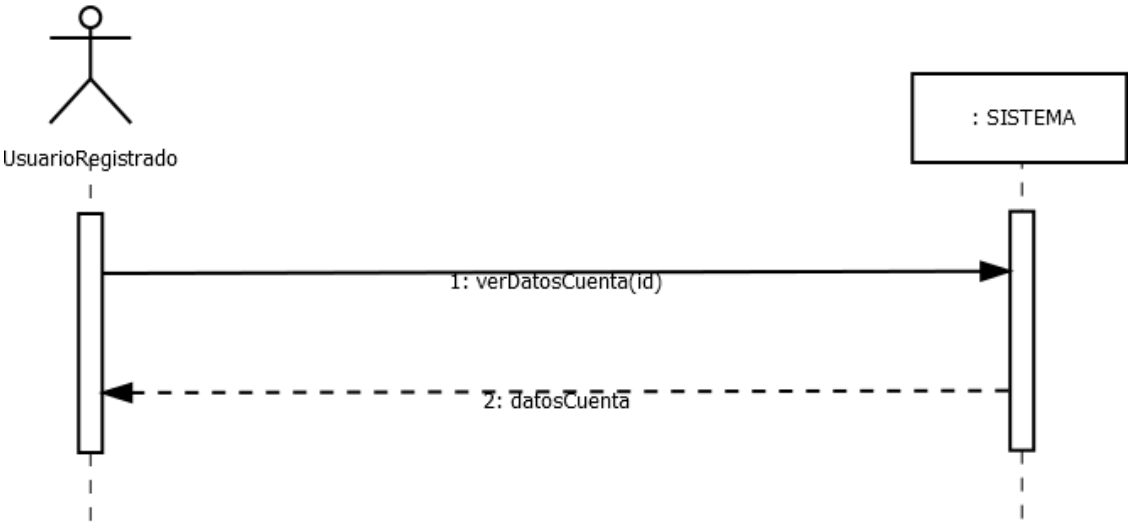


Figura 6. *Esquema ver los datos de la cuenta.*

5. Actualización cuenta.

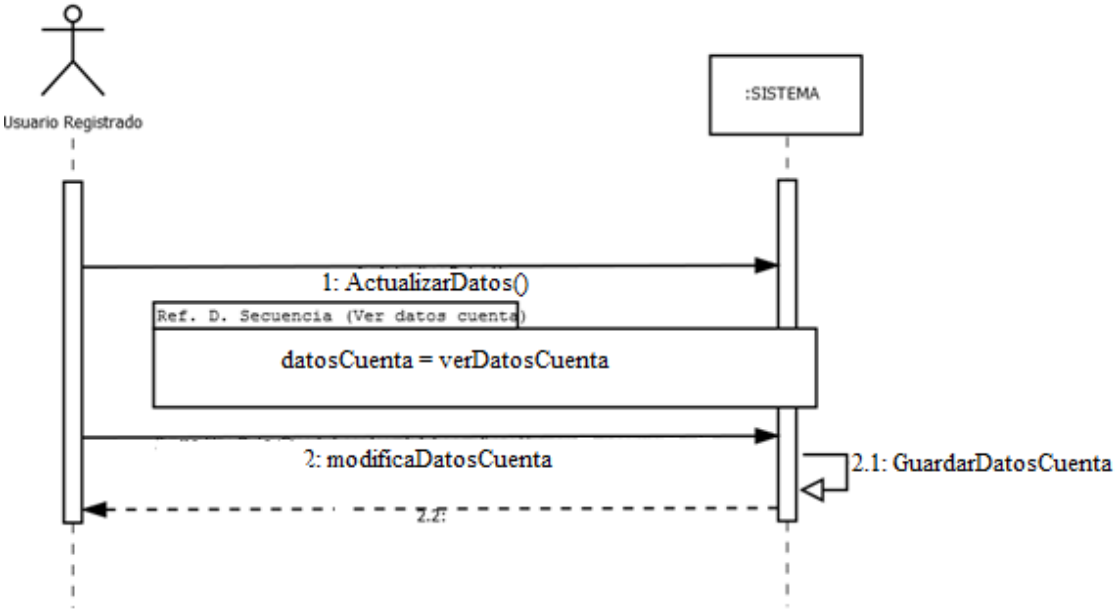


Figura 7. *Esquema de la actualización de la cuenta.*

6. Desactivación cuenta.

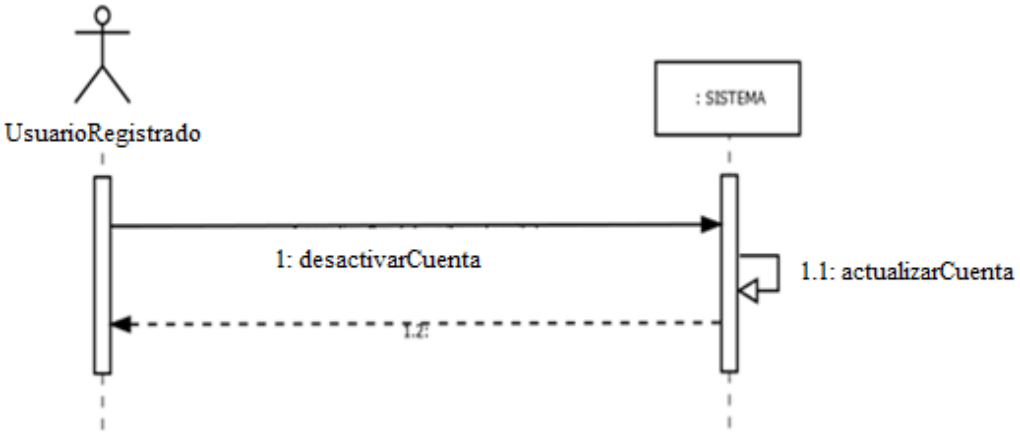


Figura 8. *Esquema de la desactivación de la cuenta.*

7. Recuperación contraseña.

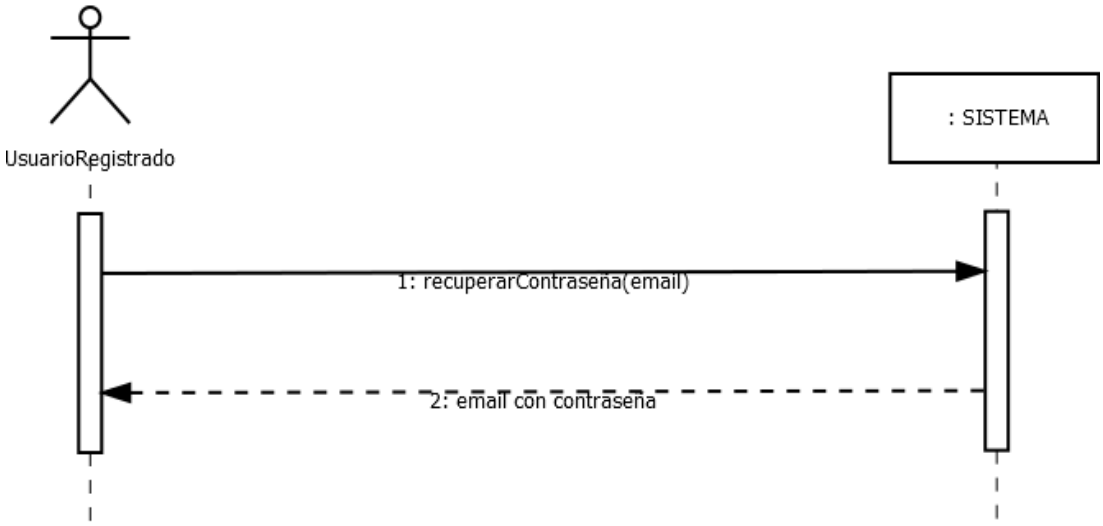


Figura 9. *Esquema de la recuperación de la contraseña.*

8. Creación evento.

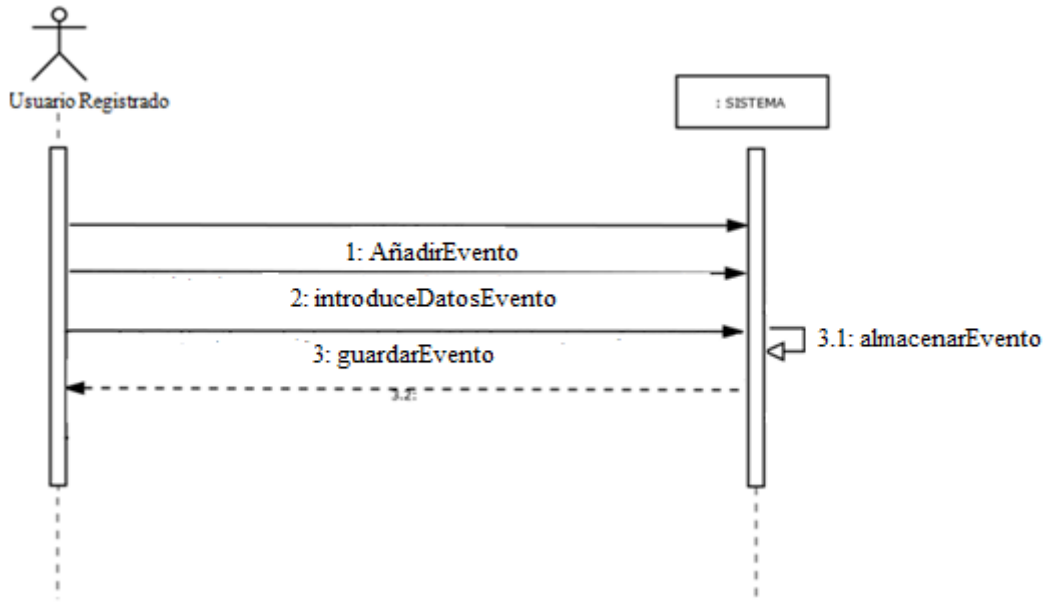


Figura 10. *Esquema de la creación del evento.*

9. Ver datos evento

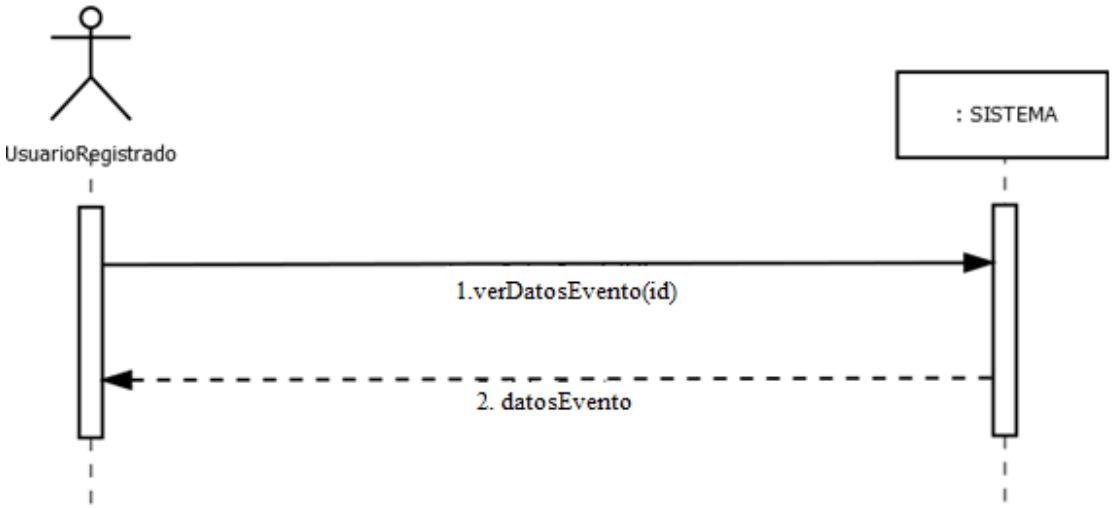


Figura 11. *Esquema ver datos del evento.*

10. Actualización evento.

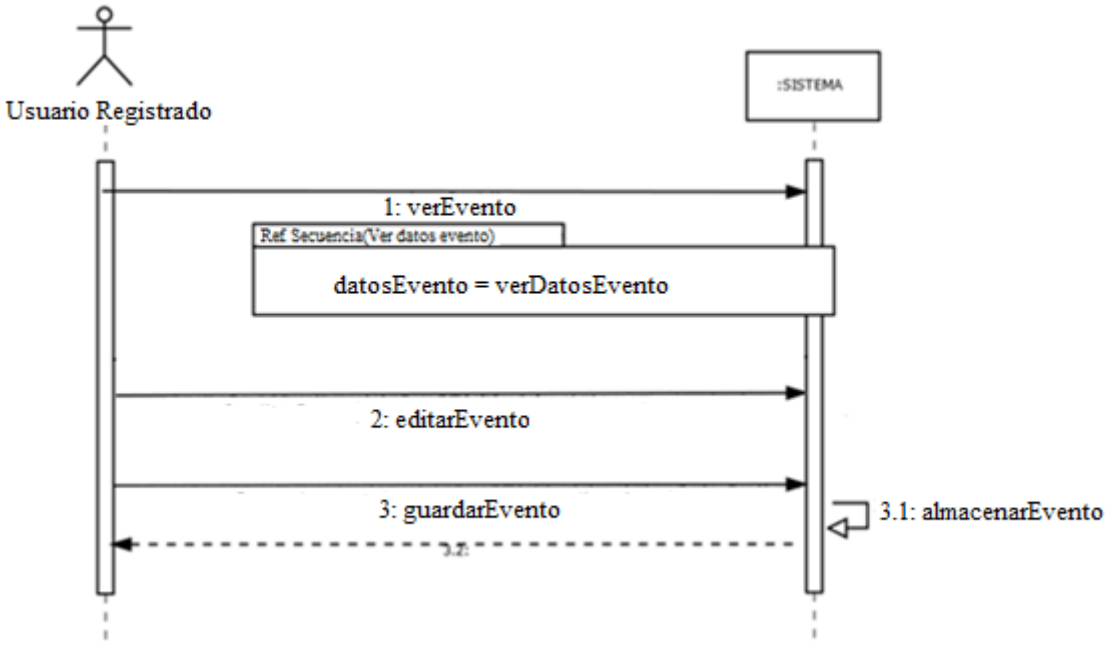


Figura 12. *Esquema de la actualización del evento.*

11. Cancelar evento.

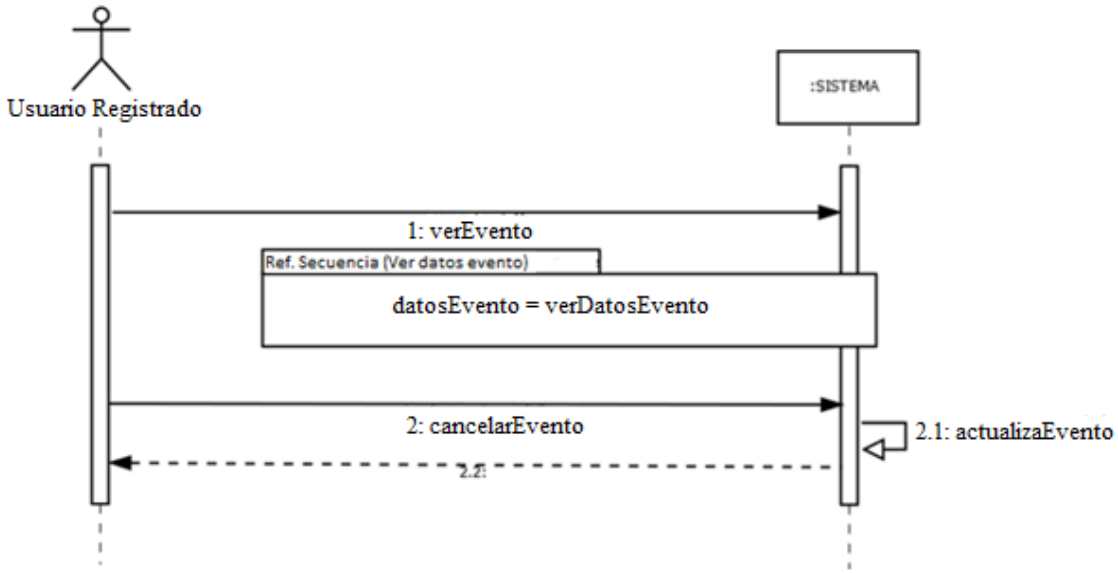


Figura 13. *Esquema de cancelación de evento.*

12. Creación incidencia.

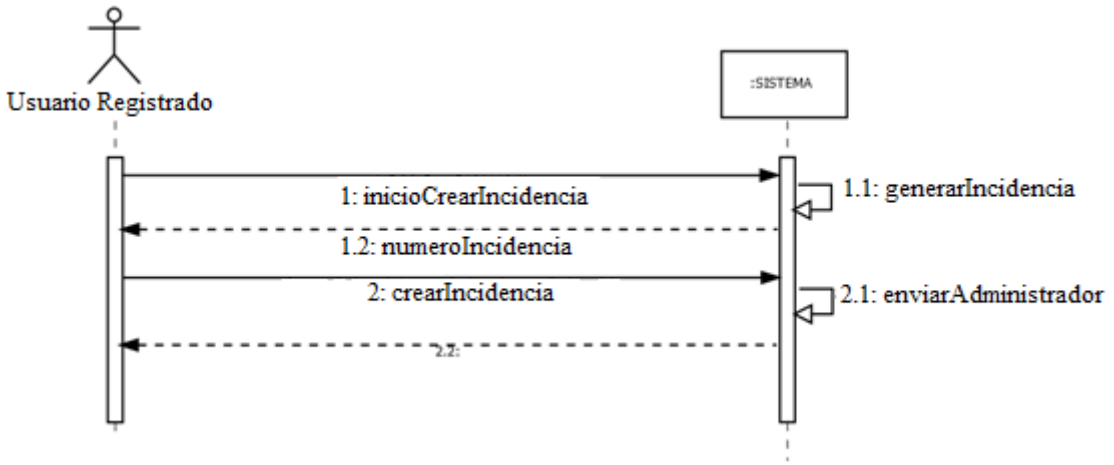


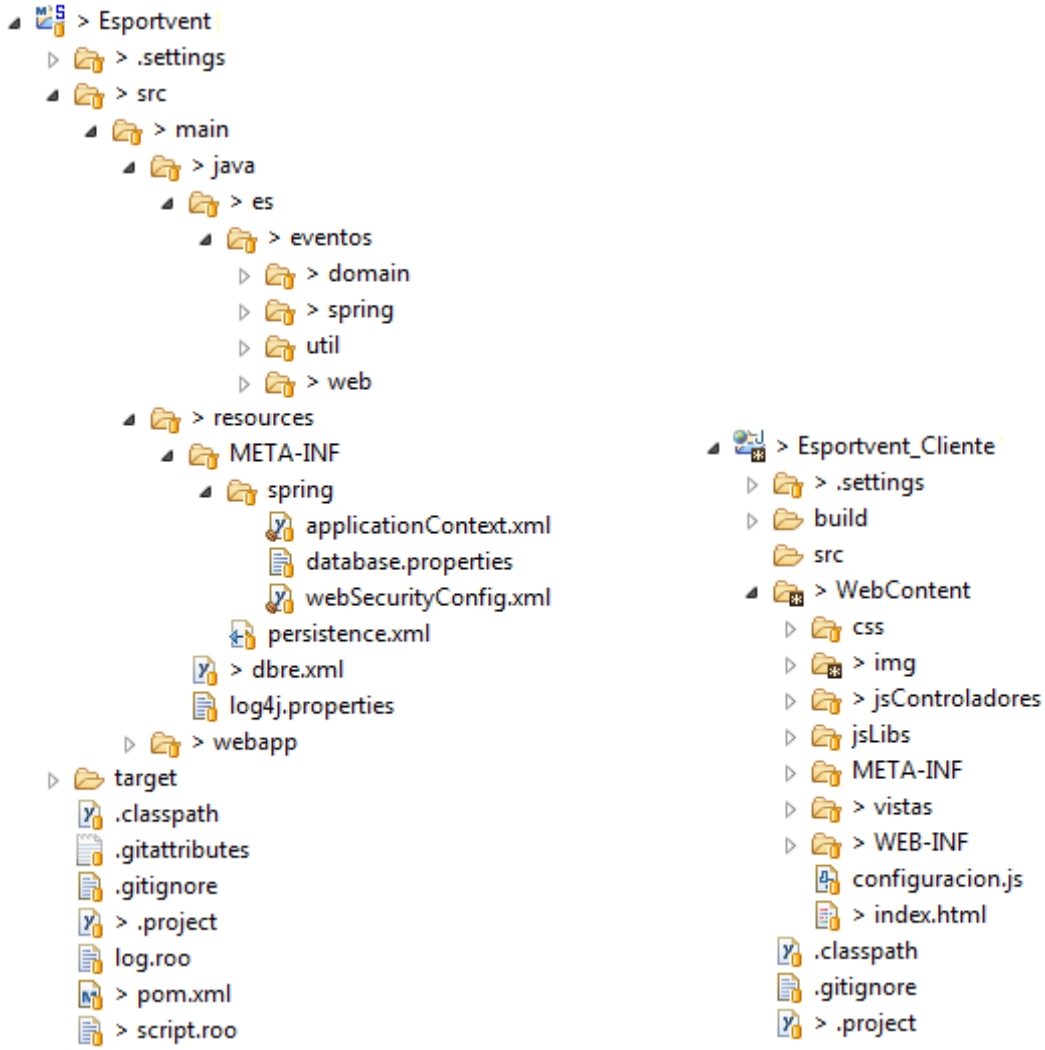
Figura 14. *Esquema de la creación de una incidencia.*

Capítulo 5. Diseño e implementación

En esta apartado se va a realizar especial hincapié en el modelo semántico de la aplicación, teniendo en cuenta la base de datos y las clases que se han usado en el desarrollo de la misma. A su vez, si finalizará con las pruebas que se le ha realizado al software.

El diseño de este proyecto se basa en una arquitectura **cliente-servidor**, no se va a entrar en más detalles ya que se puede estudiar dicha arquitectura en la Sección 5.5 (Modelado de Sistema) de este documento.

La **implementación** de la aplicaciones cliente y servidora utilizan diferentes tecnologías qué se especifican con detalle en el Capítulo 2 (Tecnologías y herramientas utilizadas) de éste documento. La estructura de los proyectos es la que se muestra en la Figura 15.



a) Aplicación servidor b) Aplicación cliente

Figura 15. *Estructura de las aplicaciones servidor y cliente*

5.1. Identificación de entidades.

En primer lugar identificaremos las entidades que van a existir en nuestro modelo relacional:

- **Usuarios_registrados:** Para modelar la base de datos entenderemos por "usuario registrado" únicamente a las personas que se han registrado en el sistema, ya que son los únicos que podrán gestionar los eventos (creación, edición y eliminación).

- **Usuarios_participantes:** Los usuarios participantes son aquellos que no tiene accesos a la plataforma, pero son invitados por un usuario registrado a formar parte de una actividad iniciada por este.

- **Evento:** Cada "usuario registrado" puede tener asociado múltiples eventos, tantos como haya creado.

- **Accesos.** Será una entidad que nos permitirá registrar los usuarios con acceso al sistema y saber si han iniciado sesión en la aplicación.

- **Tipo_evento:** Definirá los posibles tipos de evento de los eventos registrados en el sistema. Como ejemplo los tipos elegidos han sido fútbol, baloncesto, pádel, vóley.

- **Estado_evento**: Definirá los posibles estados de uso de los eventos creados en el sistema.

5.2. Diagrama de entidad-relación.

En este apartado se describe el diagrama de entidad-relación donde podemos apreciar las dependencias entre entidades. Vemos también que las entidades Usuarios_Registrados y Evento son las principales en nuestro sistema. No se detallan los atributos de cada una de las entidades en el siguiente diagrama, pues ya se citaron anteriormente (ver Figura 16).

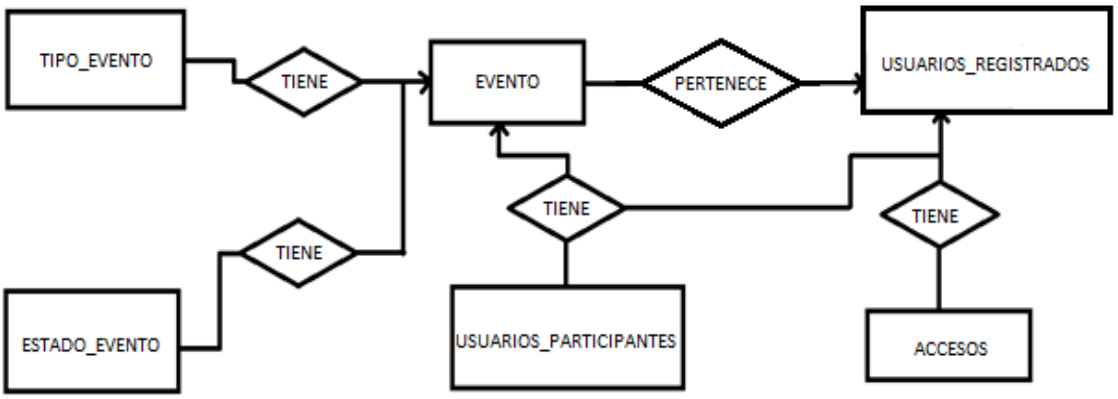


Figura 16. *Diagrama entidad-relación simplificado.*

5.3. Diagrama del modelo de base de datos.

En este apartado se va a mostrar el esquema del modelo de base de datos que se ha realizado para la aplicación en la Figura 17.

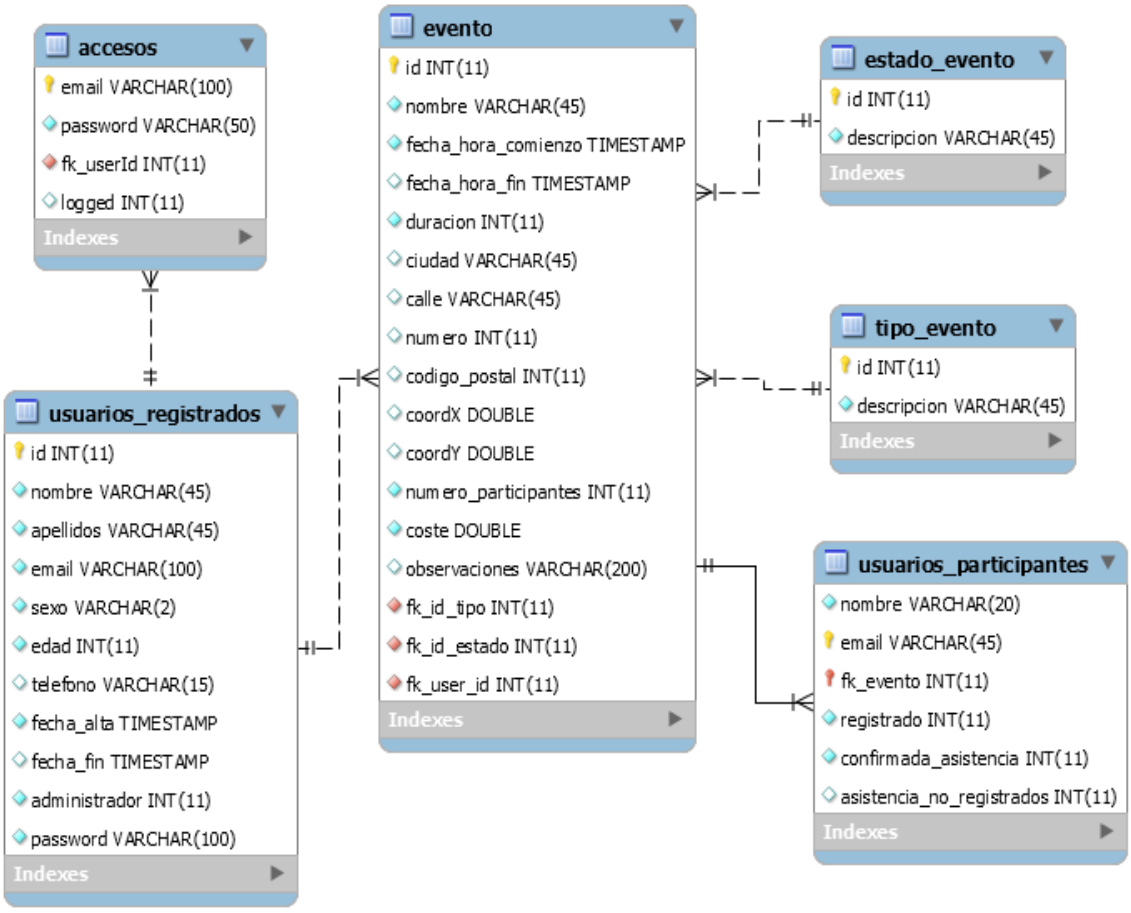


Figura 17. *Modelo de base de datos.*

5.4. Diagramas de Clases.

En esta sección se mostrarán los diagramas de clases por entidad. Como se puede apreciar se usan aspectos abstractos para extender la funcionalidad de cada entidad, esto es, conseguir un código más fácil de mantener y conseguir una separación de responsabilidades o especialización de las clases.

Todas las entidades cumplirán la misma estructura de herencia mediante aspectos. Lo explicaremos para la entidad **UsuariosRegistrados** y funcionarán de forma análoga el resto de entidades que mostraremos en los siguientes puntos de éste documento.

5.4.1. Diagrama de clases de la entidad UsuariosRegistrados.

Como vemos a continuación la clase **UsuariosRegistrados** es la entidad principal, de la que extienden una serie de clases de especialización, donde cada una de ellas tiene su propio cometido.

La clase ***Jpa_Entity** contendrá el atributo identificador de la clase UsuariosRegistrados y nos aportará los métodos de acceso de este identificador (*get* y *set*)

La clase ***DbManaged** contendrá el resto de los atributos de la clase UsuariosRegistrados y nos aportará los diferentes métodos de acceso a los atributos de la entidad, es decir, los métodos get y set de cada uno de los atributos.

La clase ***ActiveRecord** será nuestra capa de *DAO* y contendrá los métodos que implementan los accesos a datos de la tabla USUARIOS_REGISTRADOS a través de hibernate. Dichos métodos devolverán los datos requeridos de base de datos, realizarán actualizaciones, borrados e inserciones.

La clase ***Json** contendrá los métodos encargados de transformar los JSON recibidos desde la aplicación cliente en objetos Java y viceversa, transformar objetos Java en cadenas JSON para que estas puedan ser enviadas a la aplicación cliente como respuesta a las llamadas REST.

Esta arquitectura de clases es la definida comúnmente por el *framework* de Spring.

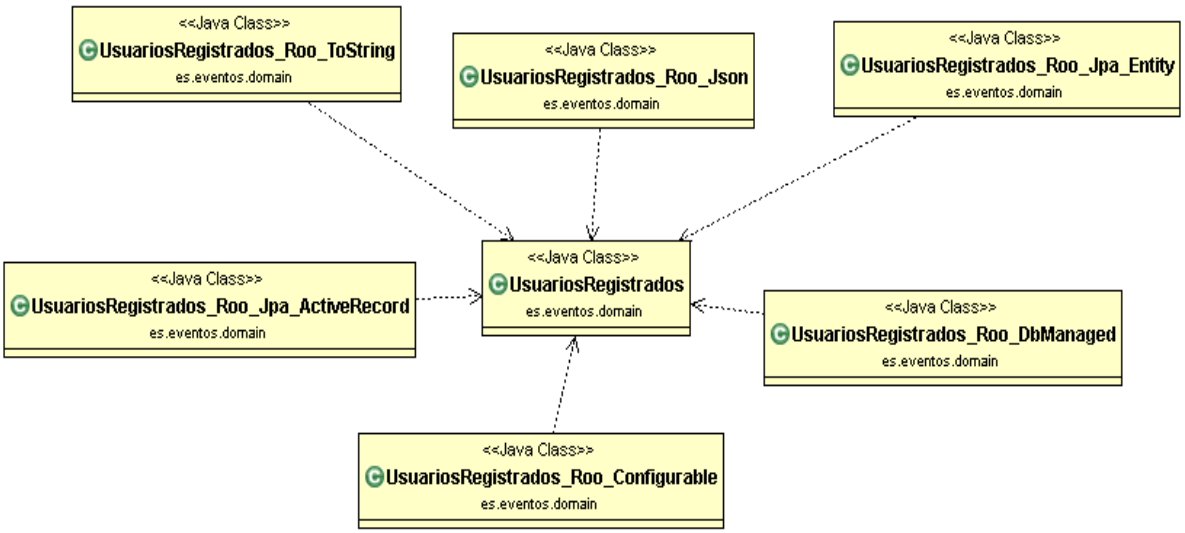


Figura 18. *Diagrama de clase de la entidad UsuariosRegistrados.*

5.4.2. Diagrama de clases de la entidad Evento.

Análogamente a la explicación de la Sección **5.4.1** podemos ver que tenemos una clase principal **Evento** de la cual extiende diferentes clases de especialización que mantienen el mismo cometido que el explicado en el citado punto.

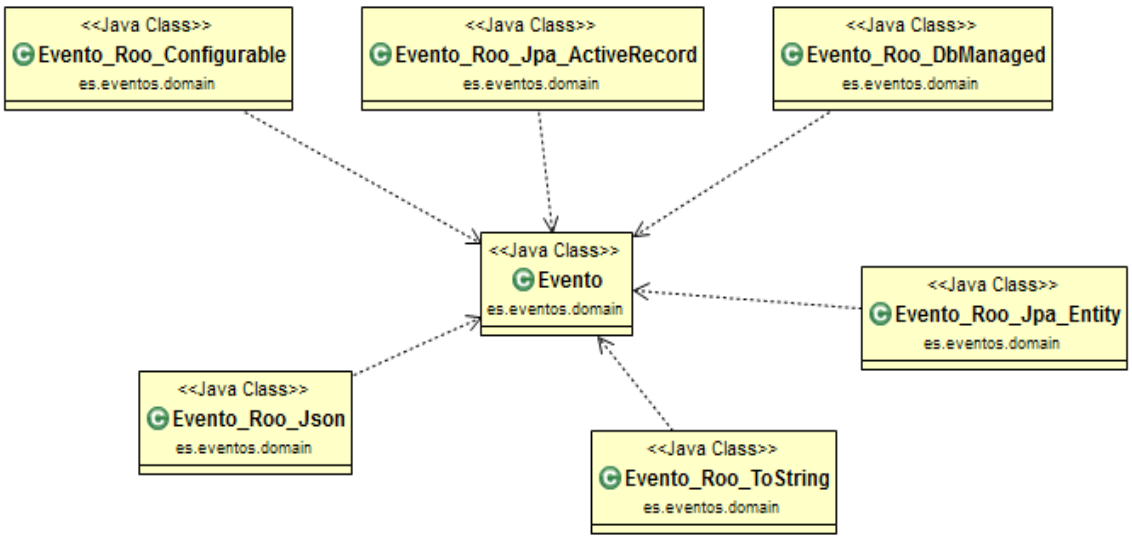


Figura 19. *Diagrama de clases de la entidad Evento.*

5.4.3. Diagrama de clases de la entidad UsuariosParticipantes.

De forma similar a los casos anteriores, podemos ver que tenemos una clase principal **Usuarios_Participantes** de la cual se extienden diferentes clases de especialización que mantienen el mismo cometido que el explicado en la Sección 5.4.1. Además se debe tener en cuenta que la tabla Usuarios_Participantes tiene una clave primaria compuesta en base de datos por lo que se usará una clase que hará referencia a esta clave compuesta.

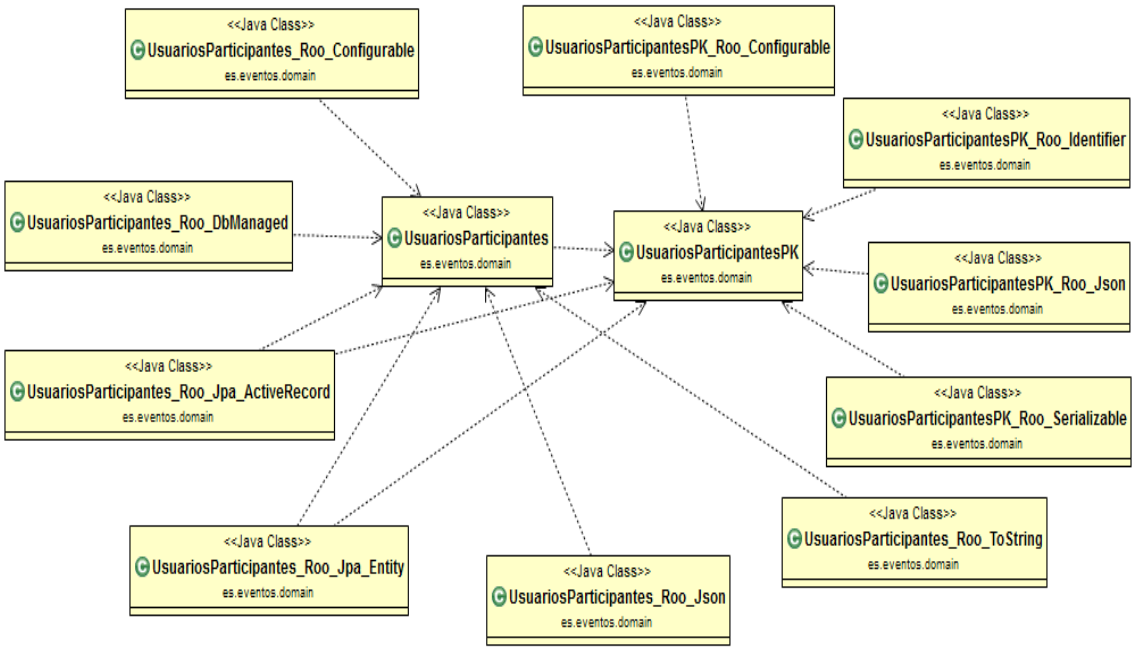


Figura 20. *Diagrama de Clases de UsuariosParticipantes.*

5.5 Modelado del sistema.

En este apartado se muestra un esquema que representa la arquitectura utilizada para la implementación del sistema completo.

En la parte superior podemos ver la arquitectura usada en el servidor y en la parte inferior la arquitectura usada en el cliente.

Como podemos observar, las comunicaciones entre la aplicación cliente y servidor se realizara mediante el uso de Web Services REST que enviaran la información en formato JSON.

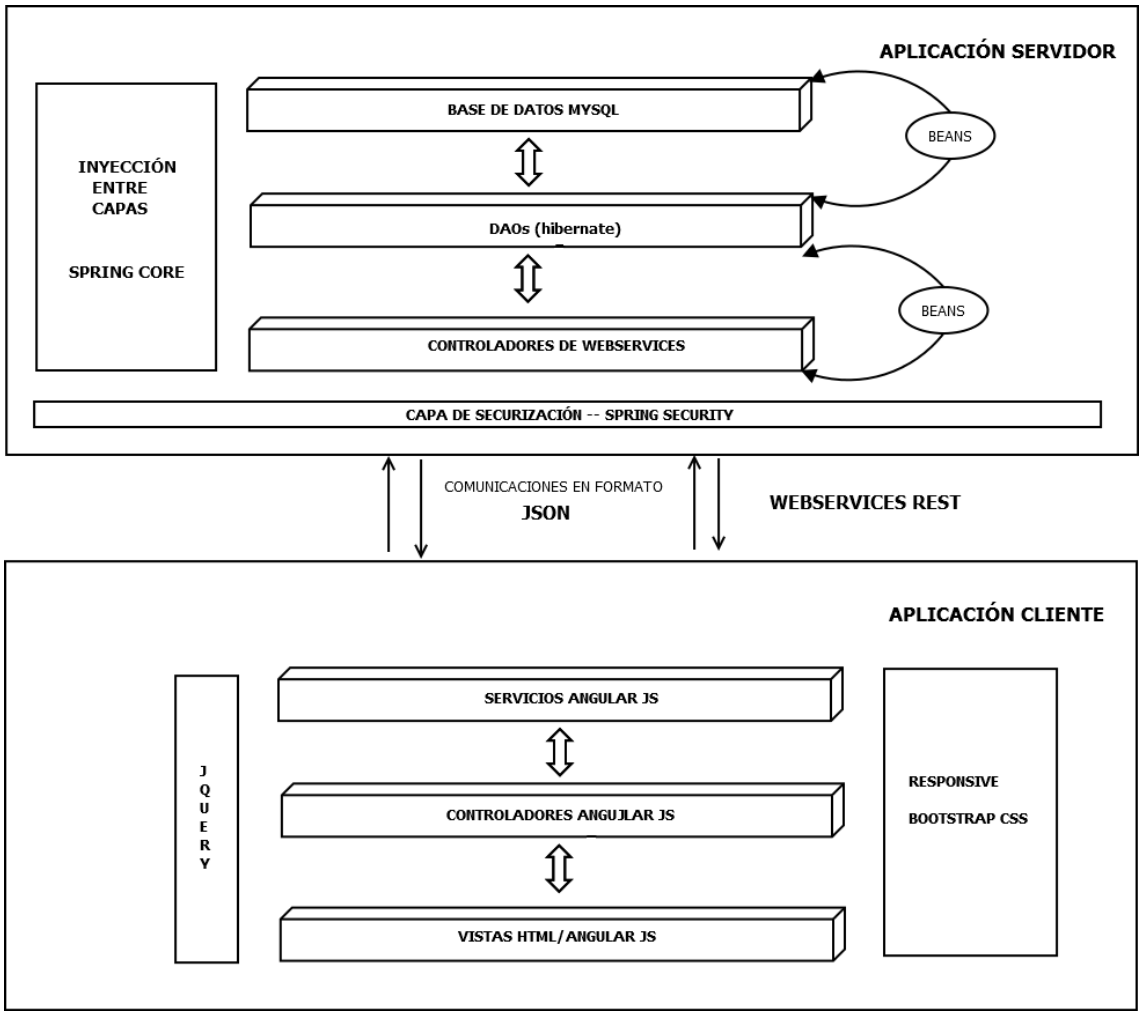


Figura 21. *Esquema aplicación servidor y cliente.*

5.6 Arquitectura de la aplicación servidor.

Si analizamos la arquitectura de la aplicación servidora podemos ver que las diferentes capas son orquestadas por Spring Core que es el encargado de manejar las inyecciones entre capas. Entre capas viajan instancias de objetos que conocemos como *beans*.

También es destacable que en la aplicación servidora se ha implantado Spring Security, lo cual nos provee de unos servicios de seguridad aportando a cada usuario *logeado* en el sistema un identificador de sesión único, de este modo Spring Security comprobará si el usuario que realiza la petición REST tiene una sesión autenticada en el sistema.

En el siguiente código (Código 1) se muestra cómo se le indica a Spring Security que métodos son los que gestionan las restricciones de acceso.

```
<!-- Entry point -->
    <beans:bean id="restAuthenticationEntryPoint"

class="es.eventos.spring.rest.api.security.RestAuthenticationEntryPoin
t" />
    <!-- Connect the custom authentication success handler -->
    <beans:bean id="mySuccessHandler"

class="es.eventos.spring.rest.api.security.RestAuthenticationSuccessHa
ndler" />
    <!-- Using default failure handler -->
    <beans:bean id="myFailureHandler"

class="org.springframework.security.web.authentication.SimpleUrlAuthen
ticationFailureHandler" />
```

Código 1. *Configuración SpringSecurity1*.

Spring Security es el encargado de capturar las llamadas a Web Services para realizar las comprobaciones anteriores mediante los métodos indicados. Mediante una configuración sencilla en XML podemos indicar que Web Services serán restringidos y cuales no (ver Código 2).

```
    <!-- Rest authentication entry point configuration -->
    <http use-expressions="true"  entry-point-
ref="restAuthenticationEntryPoint">
      <intercept-url pattern="/j_spring_security_check"
access="permitAll" />
      <intercept-url pattern="/j_spring_security_logout"
access="permitAll" />
      <intercept-url pattern="/accesoses/*"
access="hasRole('ROLE_USER')" />
      <intercept-url pattern="/accesoses" access="permitAll" />
        <intercept-url pattern="/usuariosregistradoses/*"
access="hasRole('ROLE_USER')" />
        <intercept-url pattern="/usuariosregistradoses"
access="permitAll" />
        <intercept-url pattern="/eventoes/*"
access="hasRole('ROLE_USER')" />
        <intercept-url pattern="/eventoes"
access="hasRole('ROLE_USER')" />
        <intercept-url pattern="/eventoes*"
access="hasRole('ROLE_USER')" />

        <sec:form-login authentication-success-handler-
ref="mySuccessHandler"
            authentication-failure-handler-ref="myFailureHandler" />

        <logout />
    </http>
```

Código 2. *Configuración SpringSecurity2*.

Otro aspecto relevante es que existe una capa de Controladores de Web Services que será la encargada de recibir las llamadas a Web Services y que implementará toda la lógica de negocio, haciendo uso de la capa de *DAO* para el acceso a base de datos.

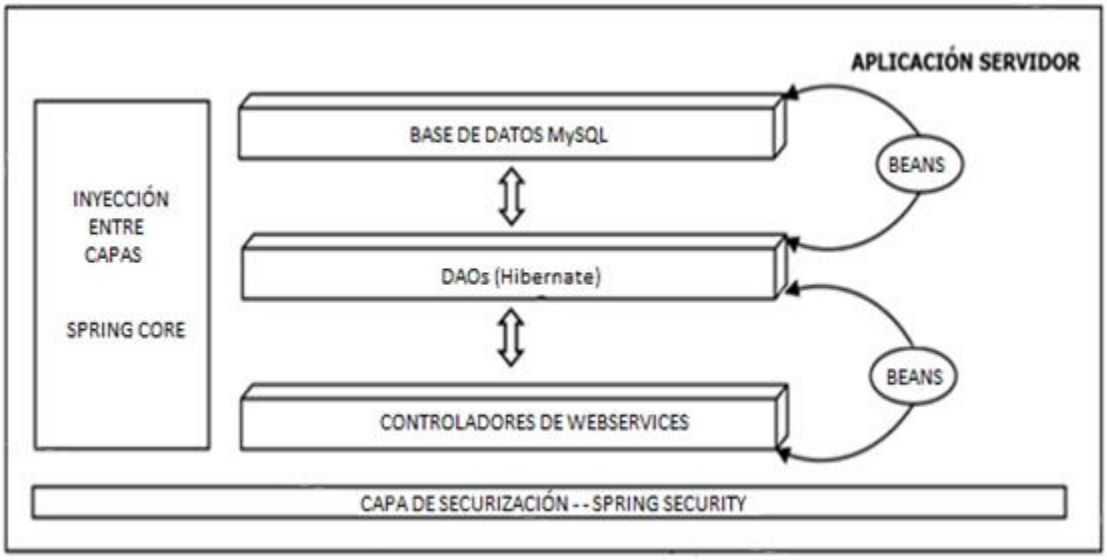


Figura 22. *Esquema aplicación servidor.*

En los siguientes apartados se detallan diagramas de clases de la capa de "Controladores de Web Services".

5.6.1. Diagrama de Clases del controlador de UsuariosRegistrados.

Como vemos en la siguiente figura, el controlador **UsuariosRegistradosController** contiene la implementación de los diferentes Web Services REST que se publican para la entidad **UsuariosRegistrados**.

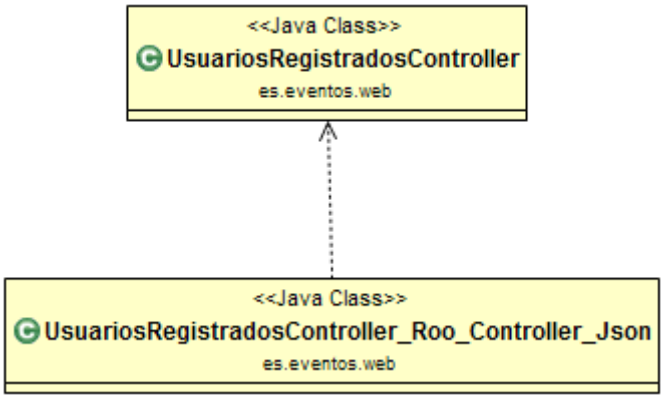


Figura 23. *Diagrama de clases del controlador de UsuariosRegistrados.*

Al igual que ocurría con la implementación de las entidades se utiliza una estructura de herencia mediante aspectos, quedando los métodos que sirven los Web Services totalmente desacoplados. Es decir, si en un futuro lo deseamos podríamos crear otros métodos diferentes que consuman un formato de información que no sea JSON o que implemente una funcionalidad diferente, estos nuevos métodos estarían definidos en una clase independiente que también heredaría de UsuariosRegistradosController. Actualmente podemos ver como todos los métodos consumen y devuelven información con formato JSON.

5.6.2 Diagrama de Clases del controlador de Evento.

De manera análoga a la Sección **5.6.1**, podemos ver el diagrama del controlador **EventoController.**

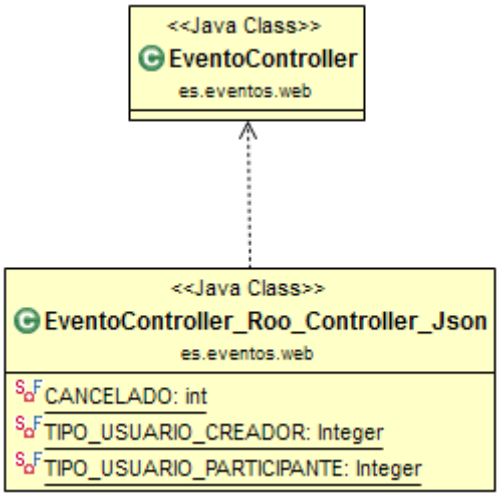


Figura 24. *Diagrama de clases del controlado de Evento.*

5.6.3 Diagrama de Clases del controlador de UsuariosParticipantes.

De manera análoga al punto **5.6.1**, podemos ver el diagrama del controlador **UsuariosParticipantesController**.

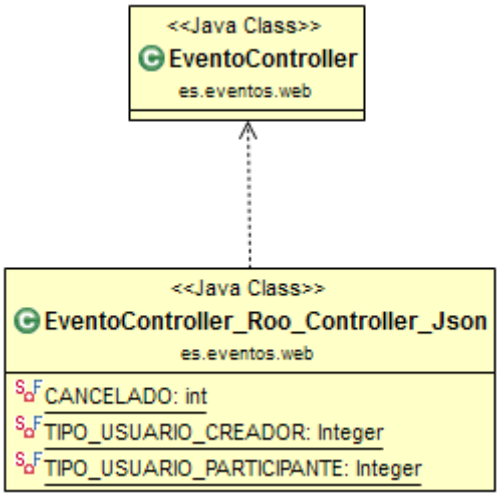


Figura 25. *Diagrama de clases del controlador UsuariosParticipantes.*

5.7. Arquitectura de la aplicación cliente.

Si analizamos la arquitectura de la aplicación cliente podemos ver que es una arquitectura basada en un cliente ligero en JavaScript. El objetivo es que podamos ejecutar la aplicación en cualquier navegador de un terminal móvil de manera rápida y eficiente.

Para orquestar dicha arquitectura se va a utilizar el *framework* de JavaScript AngujarJS. Este *framework* nos provee de un modelo de implementación que nos separa claramente una capa de servicios, donde realizaremos las llamadas a los Web Services, y una capa de controladores, donde realizaremos la lógica de negocio en el cliente. Por ejemplo, en la última capa validaremos formularios de la vistas, trataremos los objetos recibidos desde la aplicación servidora, realizaremos llamadas a las APIs de geo posicionamiento de Google [14], etc.

Por último implementaremos una capa de vistas HTML donde utilizaremos directivas propias de AngularJS, que nos facilitará la carga de objetos en formularios, validaciones de éstos, mensajes de error, etc.

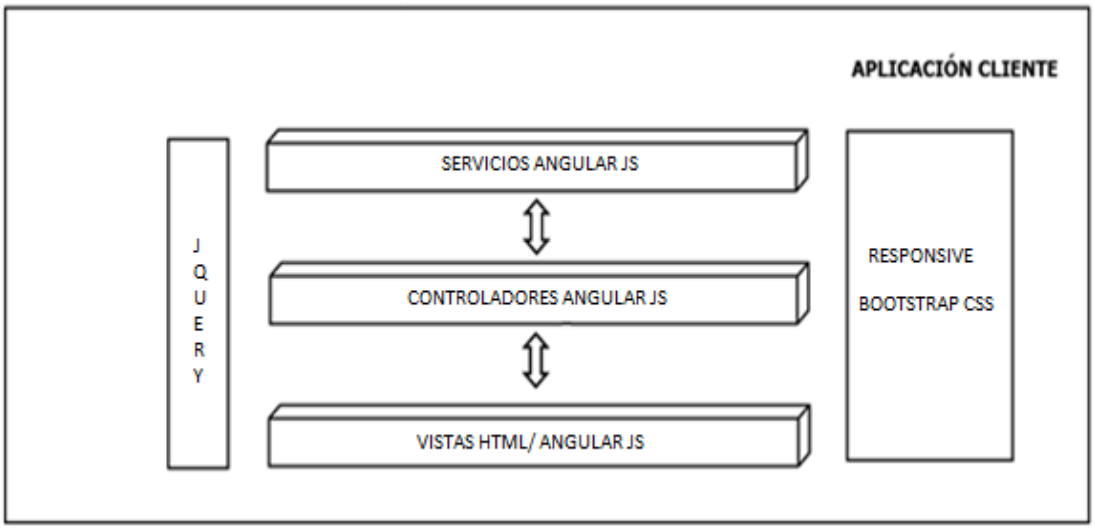


Figura 26. *Esquema de la aplicación cliente.*

Además, todo lo anterior es complementado con JQuery ya que AngujarJS hace uso de esta potente biblioteca de JavaScript.

Por ultimo añadir que se utiliza CSS y Bootstrap, para permitir que la aplicación sea *responsive* y se adapte adecuadamente a los diferentes tamaños de pantalla de los dispositivos.

La arquitectura en AngularJS es definida a través del fichero JavaScript app.js, donde se definen las diferentes rutas de navegación, las vistas que se corresponden con cada ruta y los controladores asociados a cada una de las vistas (ver Código 3).

```
'use strict';

// Declare app level module which depends on filters, and servicios
angular.module('sportEvent', ['sportEvent.filtros',
'sportEvent.servicios', 'sportEvent.directivas',
'sportEvent.controladores']).
    config(['$routeProvider', function ($routeProvider) {

$routeProvider.when('/registro-usuario', {templateUrl:
'vistas/registro-usuario.html', controller: 'CreacionUsuarioCtrl'});

$routeProvider.when('/acceso', {templateUrl: 'vistas/acceso.html',
controller: 'AccesoCtrl'});

$routeProvider.when('/seleccion-operacion/:id', {templateUrl:
'vistas/seleccion-operacion.html', controller:
'SeleccionOperacionCtrl'});

$routeProvider.when('/detalle-usuario/:id', {templateUrl:
'vistas/detalle-usuario.html', controller: 'DetalleUsuarioCtrl'});

$routeProvider.when('/creacion-evento/:usuarioId', {templateUrl:
'vistas/creacion-evento.html', controller: 'CreacionEventoCtrl'});

$routeProvider.when('/detalle-evento/:id', {templateUrl:
'vistas/detalle-evento.html', controller: 'DetalleEventoCtrl'});

$routeProvider.when('/geoLocalizacion/:usuarioId', {templateUrl:
'vistas/geoLocalizacion.html', controller: 'GeoLocalizacionCtrl'});

$routeProvider.when('/acerca-de', {templateUrl: 'vistas/acerca-
de.html', controller: 'AcercaDeCtrl'});

$routeProvider.when('/creacion-incidencia/:usuarioId', {templateUrl:
'vistas/creacion-incidencia.html', controller: 'IncidenciaCtrl'});

$routeProvider.when('/restaurar-contrasena', {templateUrl:
'vistas/restaurar-contrasena.html', controller: 'ContrasenaCtrl'});

$routeProvider.when('/listar-eventos-creados/:usuarioId',
{templateUrl: 'vistas/listar-eventos-creados.html', controller:
'ListarEventoCtrl'});

$routeProvider.when('/listar-eventos-participantes/:usuarioId',
{templateUrl: 'vistas/listar-eventos-participantes.html', controller:
'ListarEventoParticipantesCtrl'});

$routeProvider.when('/error', {templateUrl: 'vistas/error.html',
controller: 'ErrorCtrl'});

$routeProvider.otherwise({redirectTo: '/acceso'});

    }]);
```

Código 3. *Arquitectura AngularJS.*

La relación entre los controladores y los servicios es definida en los propios controladores, donde se instancian los servicios.

5.8 Pruebas unitarias

Para comprobar que el sistema funciona de manera correcta se han realizado diferentes pruebas en el sistema que lo verifican.

- Registro en el sistema.
  El usuario introduce los campos obligatorios en el formulario de registro de la aplicación y los guarda.
  Resultado: satisfactorio.

- Acceso a la aplicación.
  Una vez registrado, el usuario introduce su correo electrónico y contraseña en la pantalla de acceso.
  Resultado: satisfactorio.

- Actualización de datos usuario.
  El usuario accede a la pantalla de detalles de usuario. Comprueba que los datos son correctos y, si lo desea, actualiza los datos del formulario.
  Resultado: Satisfactorio.

- Creación evento.
  El usuario accede a la pantalla de crear evento, rellena los campos obligatorios del formulario, entre los que se encuentran los participantes del evento, y se guarda correctamente.
  Resultado: satisfactorio

- Participación en evento:
  El usuario confirma o rechaza la asistencia al evento en el que ha sido invitado a participar.
  Resultado: satisfactorio

- Eliminar cuenta.
  El usuario elimina su cuenta de usuario, mostrando un mensaje informando al usuario.
  Resultado: satisfactorio

- Creación incidencia.
  El usuario avisa al administrador del sistema de la posible incidencia mediante correo electrónico.
  Resultado: satisfactorio

- Fin del acceso.
  El usuario se desconecta del sistema, volviendo a la pantalla de acceso.
  Resultado: satisfactorio

Capítulo 6. Conclusiones

En este último capítulo se va a comentar en las conclusiones obtenidas durante el desarrollo de este Trabajo Final de Grado junto con las posibles mejoras que se podrían aplicar a la aplicación.

6.1 Resultado final y conclusiones

En este TFG se ha propuesto el desarrollo de una aplicación móvil para poder gestionar eventos deportivos. Durante el desarrollo de este proyecto se han usado diferentes tecnologías, tal y como se ha expuesto en capítulos anteriores. Se han elegido estas tecnologías porque nunca había trabajado con ellas y me parecían muy interesantes, a la vez que motivador, desarrollar con estas herramientas. Debido a ello, he mejorado mis conocimientos en AngularJS, Web Services REST, JSON y el resto de herramientas usadas en este proyecto.

Ha sido una experiencia muy positiva debido a que he tenido que adquirir ciertos conocimientos referentes a estas nuevas tecnologías y pienso que puede ser muy beneficioso para mi futuro. El aprendizaje ha sido duro, debido a que la mayoría de tecnologías usadas eran nuevas para mí y he tenido que esforzarme para conseguir los resultados deseados. La parte positiva es que hay muchísima información en la red y es relativamente sencillo encontrar materiales o soluciones para el desarrollo de la aplicación.

Una de mis principales aficiones es el deporte, por lo que desarrollar una aplicación en la que puedes facilitar la organización de eventos deportivos con amigos me pareció de gran utilidad para aquellas personas que practican deportes de equipo, a la vez que un gran reto por la complejidad que conlleva.

Antes de comenzar con el desarrollo de la aplicación había usado plataformas similares, como Rosterbot o Padelon. La idea era elaborar una aplicación con la que los usuarios pudiesen organizar un evento de manera fácil, sencilla y cómoda. Por ello se ha realizado una interfaz amigable y un sistema de mensajes por correo electrónico para que los usuarios puedan confirmar o rechazar la asistencia al mismo. Además se ha dotado a la plataforma de un sistema de cálculo de coordenadas para realizar la geolocalización de la posición del usuario con el evento, de esta manera los usuarios podrán utilizar el navegador de su dispositivo para llegar al evento.

Para esta parte del proyecto tuve que conseguir diferentes "*KEYS*" usadas por la compañía Google que permiten la apertura de la aplicación "Google Maps" [15] y así poder utilizar la aplicación como GPS.

6.2 Trabajos futuros

Es una aplicación muy útil para aquellas personas que deseen organizar eventos relacionados con el deporte con sus amigos, pero aun así podrían plantearse las siguientes mejoras:

- Los eventos podrían ser de tipo "público". Actualmente sólo son de tipo privado por lo que únicamente pueden asistir aquellas personas cercanas al organizador del evento. Si se realizasen eventos de tipo público, todos los usuarios registrados que lo desearan, al menos hasta completar los participantes en el evento, podrían asistir a este tipo de eventos.
- Mensajes por aplicaciones de sistemas de mensajería instantánea. En la aplicación se avisa a los usuarios por correo electrónico.
- Instalación de servicio de chat. De esta manera los participantes en un evento podrían estar comunicados y así conocer las posibles modificaciones en los eventos en los que participen.

APÉNDICES

I. MANUAL DE USUARIO

Este apartado es una guía de utilización para usuarios, con esta guía no será necesario que los usuarios tengan conocimientos previos para usar Esportvent.

A continuación se reflejarán una serie de puntos donde se explicarán las diferentes funcionalidades que soporta la aplicación Esportvent.

I.1 Modo de uso

Cualquier usuario, sin necesidad de estar registrado podrá acceder a esta opción, desde la pantalla de *login* de la aplicación debemos pulsar el icono de información  que nos llevará a esta pantalla, donde se enumeran todas las posibilidades que nos aporta la aplicación:

Figura 27. *Modo de funcionamiento.*

I.2 Registro de usuario

Para darse de alta en el sistema simplemente es necesario pulsar el botón 

de la pantalla de *login*, a raíz de esta pulsación se mostrará el formulario de alta:

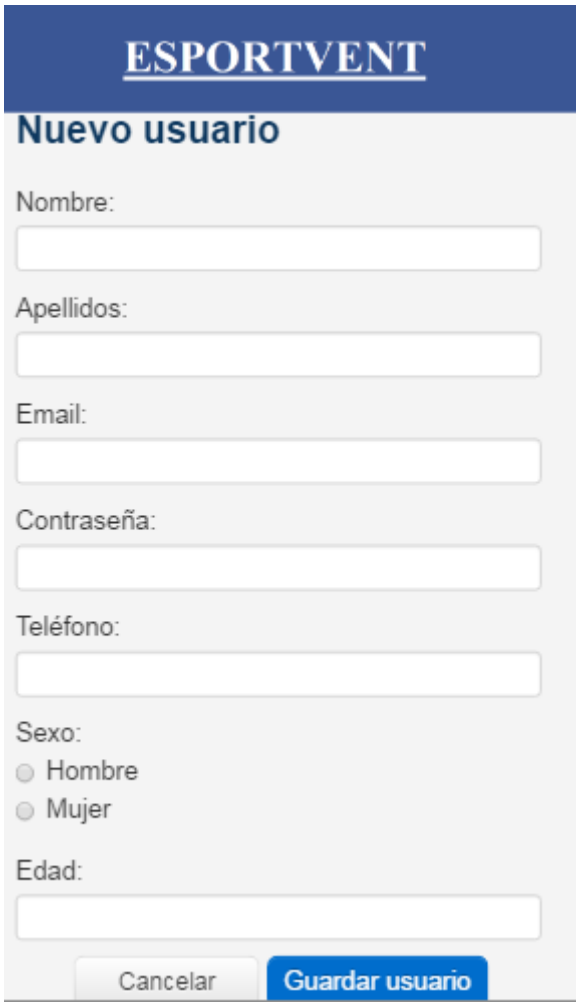


Figura 28. *Alta de usuario.*

Todos los campos de este formulario son obligatorios, y en cada uno de ellos se validará el formato y el tamaño de la información, en caso de introducirla incorrectamente.

I.3 Autenticación de usuarios

Para acceder a las secciones restringidas del sistema es necesario que el usuario se autentique, para ello, es necesario que el usuario introduzca el email y la contraseña que utilizó para darse de alta en el sistema, de este modo, podrá acceder a las opciones restringidas:

Figura 29. *Control de acceso.*

I.4 Selección de operaciones

Una vez nos hemos accedido al sistema se nos mostrará la pantalla "Selección de opciones", en esta pantalla se nos listan las diferentes operativas que podemos realizar estando autenticados en el sistema, entre ellas "actualizar datos de usuario", "crear evento", "asistencia a eventos", etc.

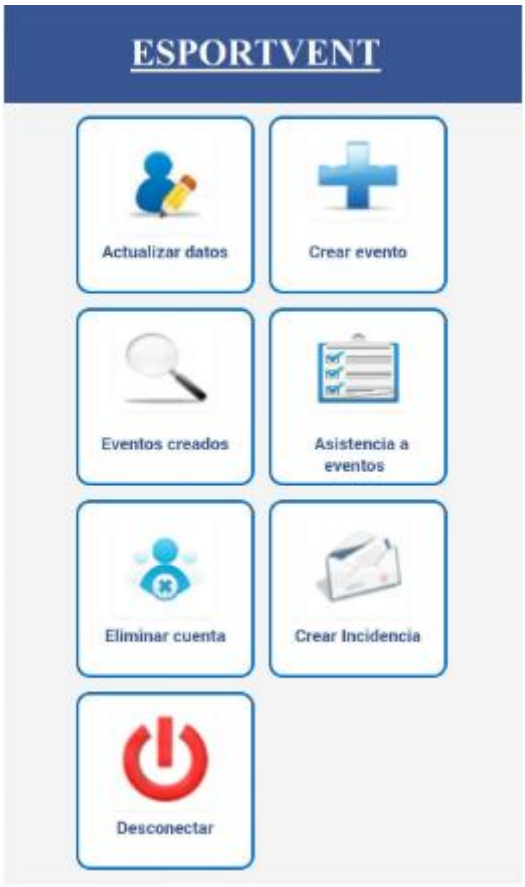


Figura 30. *Selección operación.*

I.5 Actualización de datos

Desde la pantalla de selección de operativa se puede elegir la opción, a través de la cual se nos mostrará un formulario similar al de registro, pero con los datos de usuario precargados, este formulario nos permitirá actualizar cualquier información del usuario que se desee.

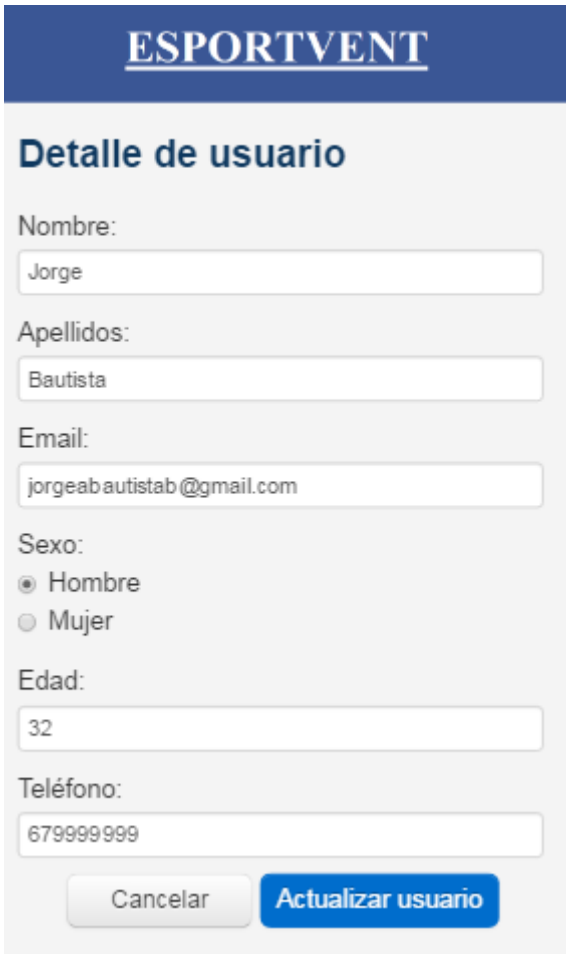


Figura 31. *Actualización datos usuario.*

I.6 Crear evento

Desde la pantalla de selección de operativa se puede elegir la opción , a través de la cual se puede acceder a la operativa que nos permitirá crear un evento para la gestión del mismo:

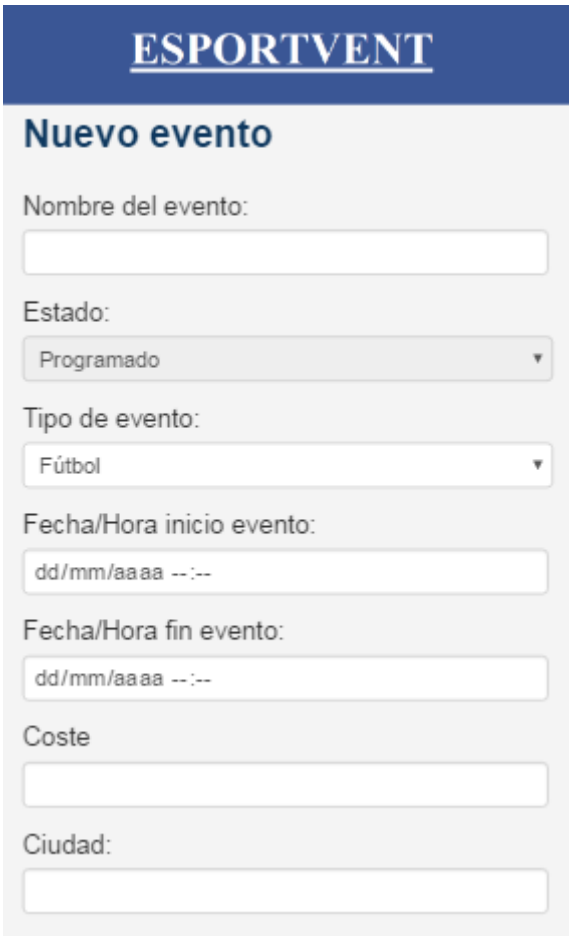


Figura 32. *Creación de un evento.*

Se solicitarán los diferentes datos del evento, nombre, coste, ciudad, calle, etc. Una vez rellenos los datos de dirección se establecerán de forma automática las coordenadas del evento. Si las coordenadas no se pueden obtener el evento no podrá darse de alta.

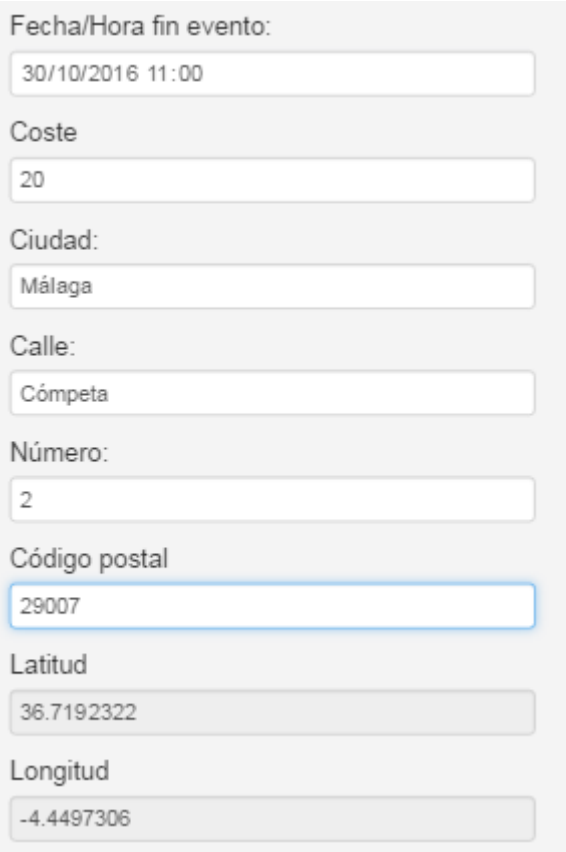


Figura 33. *Dirección del evento.*

Una vez dado de alta un evento, se podrá consultar, editar o cancelar pulsando la opción que se encuentra en la pantalla "selección-operación"

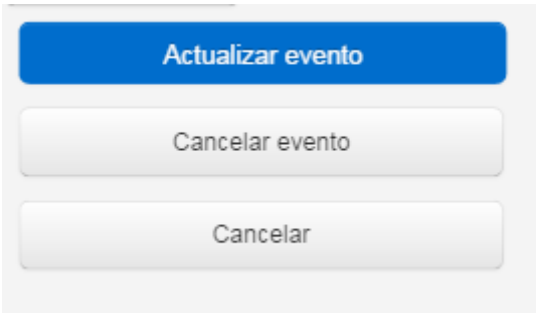


Figura 34. *Botonera actualización evento.*

I.7 Desactivar cuenta

Desde la pantalla de selección de operativa se puede elegir la opción, a través de la cual desactivaremos de forma voluntaria nuestra cuenta, es necesario tener en cuenta que aunque se desactive la cuenta, los eventos asociados a ella seguirán almacenados en la base de datos.

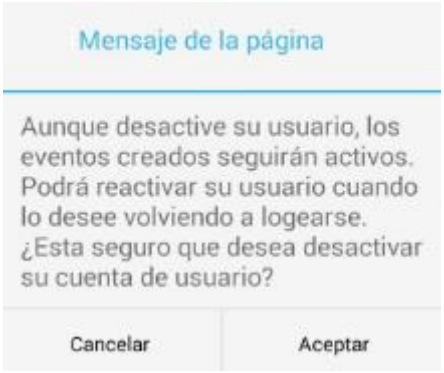


Figura 35. *Mensaje desactivación de usuario.*

Cualquier cuenta puede ser reactivada cuando se desee tras una desactivación. Mientras que la cuenta esté desactivada no podremos acceder a las zonas restringidas de la aplicación.

I.8 Creación de incidencia

Desde la pantalla de selección de operativa se puede elegir la opción, a través de la cual podremos enviar una incidencia o consulta al administrador del sistema, por ejemplo si los usuarios no han podido crear un evento correctamente.

Cada incidencia tiene un número de incidencia único que se genera de forma automática.

Figura 36. *Creación de incidencia.*

Automáticamente esta incidencia se enviará por correo electrónico al administrador.

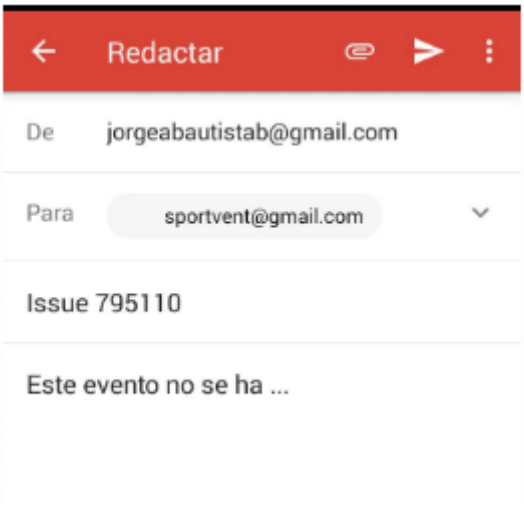


Figura 37. *Correo electrónico al administrador.*

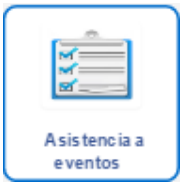
I.9 Recuperación de contraseña

Desde la pantalla de login podemos pulsar el enlace, a través del cual nos aparecerá un formulario donde introduciremos nuestro email. De este modo nuestra contraseña nos será enviada a nuestra cuenta de correo.

Figura 38. *Recuperación de la contraseña.*

I.10 Asistencia a eventos

Desde la pantalla de selección de operativa se puede elegir la opción , a través de la cual aparecerá un listado con los eventos en los que el usuario ha participado junto con las coordenadas de la posición actual en la que se encuentre. Cada evento tiene el nombre, estado, tipo de deporte, fechas de inicio/fin y un enlace indicando si asistirá al evento.



Cuando queramos dirigirnos a algunos de los eventos listados bastará con pulsar “IR AHORA” e inmediatamente se establecerá la ruta a seguir para llegar hasta el destino mediante la aplicación de navegación GPS que tenga predefinida el terminal.

Figura 39. *Listado de participación de eventos*.

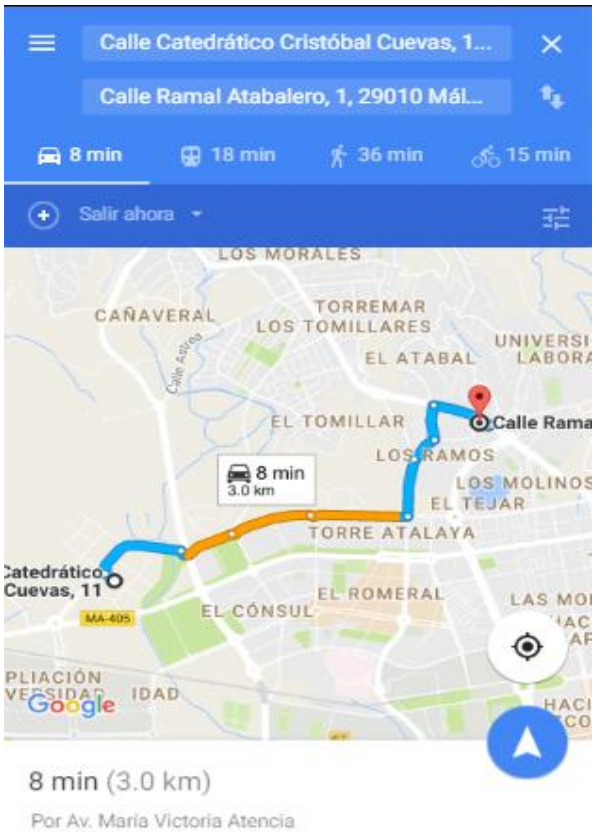


Figura 40. *Información de la ruta ofrecida por Google Maps®.*

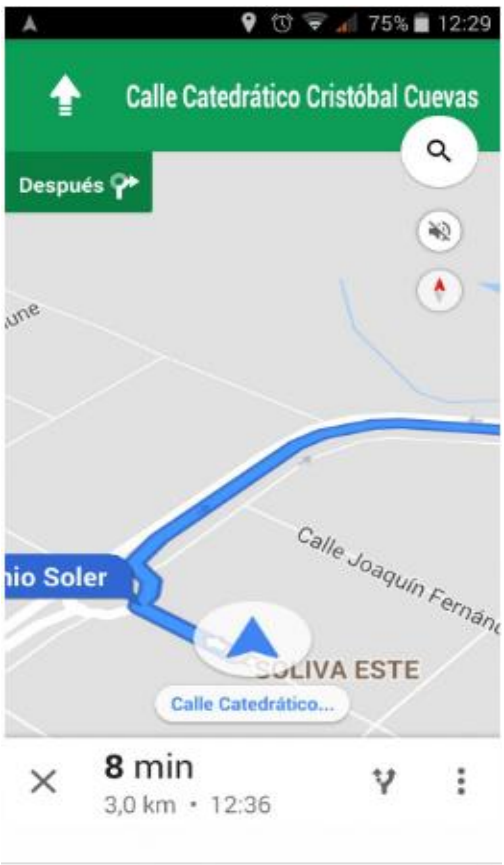


Figura 41. *Navegación al destino mediante Google Maps®.*

Al igual que se pueden listar los eventos más cercanos, desde la vista de listado se ofrece la opción, desde la cual se nos abrirá una pantalla que representará los eventos listados sobre un mapa, pudiendo pulsar sobre ellos para ser guiados mediante el GPS.

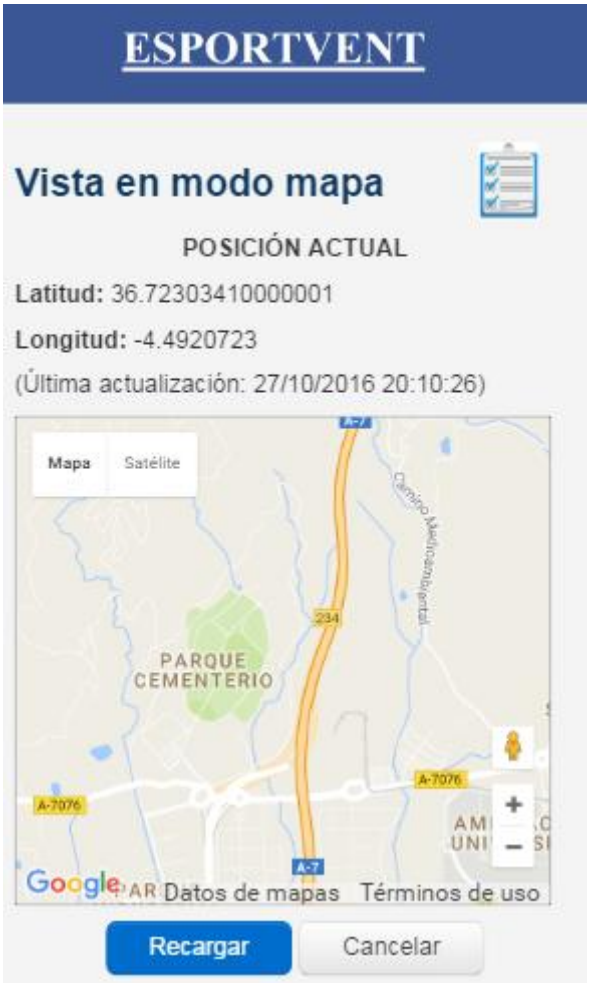


Figura 42. *Vista en modo mapa.*

Se podrá volver a la vista en modo lista por medio de la opción  .

II Manual de despliegue

II.1 Creación de base de datos.

Instalar una base de datos MYSQL y lanzar los scripts necesarios para obtener la base de datos operativa.

II.2 Despliegue de aplicaciones cliente y servidora.

Para desplegar la aplicación tan solo tenemos que instalar un servidor Tomcat 7 y desplegar en su carpeta "*webapps"*.

II.3 Configuración de los ficheros de conexión:

- **Configuración de IP donde se sirven los webservices:** Para configurar la IP y el puerto donde está desplegada la aplicación servidora se realizará en el fichero "configuration.js" (ver Codigo 4):

```
function getServerIp(){

	var ip = "http://192.168.1.146:8080\:8080";

	return ip;
}
```

Código 4. *Fichero configuration.js*

Configuración de la conexión de la aplicación servidora con la Base de Datos: Es necesario configurar el fichero "database.properties", para ello será necesario configurar la url de conexión con la base de datos, y las credenciales de conexión (ver Código 5):

```
#Updated at Wed Jul 20 19:51:53 CEST 2016
#Wed Jul 20 19:51:53 CEST 2016
database.driverClassName=com.mysql.jdbc.Driver
database.url=jdbc\:mysql\://localhost\:3306/esquema_eventos
database.username=root
database.password=root
```

Código 5. *Fichero database.properties*

Tras realizar estas configuraciones, se debe iniciar el servidor. Para ello, se debe ejecutar el fichero "Tomcat7.exe", y de esta manera el tomcat arranca y despliega las aplicaciones correctamente.

II.4. Acceso a la aplicación.

Por último solo tenemos que acceder al navegador del teléfono móvil, estando el terminal móvil conectado a la misma red que el pc donde tenemos el servidor Tomcat.

http://ip:puerto/Esportvent_Cliente

Si el servidor Tomcat está sirviendo los proyectos en una ip pública con acceso desde el exterior, no será necesario que el teléfono se encuentre conectado a la misma red que el pc servidor, esto será lo común en un entorno real.

Bibliografía

[1] Rosterbot. [Plataforma web] Disponible en http://http://www.rosterbot.com/ Accedido el 11 de Septiembre de 2016

[2] Padelon. [Aplicación móvil] Disponible en https://play.google.com/store/apps/details?id=com.padelon&hl=es Accedido el 11 de Septiembre de 2016

[3] Manual de Maven. [Recurso electrónico] Disponible en https://maven.apache.org/ Accedido el 13 de Septiembre de 2016

[4] Guía de comienzo de uso de Spring Framework. [Recurso electrónico] Disponible en: http://projects.spring.io/spring-framework/ Accedido el 14 de Septiembre de 2016

[5] Guía de comienzo de uso de Spring Security. [Recurso electrónico] Disponible en http://projects.spring.io/spring-security/ Accedido el 14 de Septiembre de 2016

[6] Manual de JavaScript. Comunidad de desarrolloweb.com. [Recurso electrónico]. Disponible en http://www.desarrolloweb.com/manuales/20/ Accedido el 12 de Septiembre de 2016

[7] Manual de HTML. Comunidad de desarrolloweb.com. [Recurso electrónico]. Disponible en http://www.desarrolloweb.com/manuales/21/ Accedido el 19 de Septiembre de 2016

[8] ¿Qué es AngularJS? Primeros pasos para aprenderlo. Blog de Tecnología. [Recurso electrónico]. Disponible en https://carlosazaustre.es/blog/empezando-con-angular-js/ Accedido el 12 de Septiembre de 2016

[9] REST. Blog de Tecnología. [Recurso electrónico]. Disponible en https://eamodeorubio.wordpress.com/2010/07/26/servicios-web-2-%C2%BFque-esrest/ Accedido el 13 de Septiembre de 2016

[10] Primeros pasos con Angular. [Recurso electrónico]. Disponible en http://www.adictosaltrabajo.com/tutoriales/angularjs/ Accedido el 15 de Septiembre de 2016

[11] ¿Qué es Bootstrap y cómo funciona?. Blog de Tecnología. [Recurso electrónico]. Disponible en http://www.arweb.com/chucherias/editorial/%C2%BFque-es-bootstrap-y-como-funciona-en-el-diseno-web.htm. Accedido el 18 de Septiembre de 2016.

[12] Introducción a CSS. Eguíluz Pérez, Javier. [Recurso electrónico]. Disponible en http://librosweb.es/css/ Accedido el 19 de Septiembre de 2016

[13] Curso de AngularJS. [Recurso electrónico]. Disponible en http://campus.codeschool.com/courses/shaping-up-with-angular-js/contents Accedido el 16 de Septiembre de 2016

[14] Obtener coordenadas de un punto (Latitud y Longitud) a partir de una dirección usando Google Maps. Blog de Iván Angulo. [Recurso electronico]. Disponible en http://ivanargulo.com/blog/obtener-coordenadas-de-un-punto-latitud-y-longitud-a-partir-de-una-direccion-usando-google-maps Accedido el 20 de Septiembre de 2016

[15] Documentación API -- Geolocation.watchPosition(). Documentación para desarrolladores Mozilla. [Recurso electrónico]. Disponible en https://developer.mozilla.org/es/docs/Web/API/Geolocation/watchPosition Accedido el 20 de Septiembre de 2016